# TAMA Modecleaner alignment error signals
# Version 1.4

G. Heinzel, K. Arai, NAO Mitaka

February 19, 2001

## 1 Introduction

The effects of mirror misorientations[1] in interferometers with three or more mirrors cannot easily be determined analytically in the general case. Therefore a MATHEMATICA program for numerical 3-D ray tracing was written. It uses geometrical optics to find the axis of the eigenmode for a given combination of reflecting and/or refracting plane and spherical surfaces. A similar program is described and printed in Appendix E3 of MPQ243. The term 'beam' in this text refers to the geometrical axis of a beam, without taking into account the transverse shape or optical phase.

## 2 The modecleaner cavities

Figure 1 shows a schematic view of the TAMA modecleaner cavity seen from above, together with the coordinate system adopted in this section.

The cavity consists of two flat mirrors ($M_a$ and $M_b$) that are separated by a relatively short distance (20 cm), and a curved mirror $M_c$ with radius of curvature $R = 15\,\mathrm{m}$, which is at a distance $L = 9.738\,\mathrm{m}$ from the (center between the) flat mirrors. The beam enters through $M_a$ and travels clockwise to $M_b$, $M_c$, $M_a$, etc.

$M_e$ is the location of the photodetector behind the end mirror (in the following called 'end detector' for short), which is at a distance of 2.08 m behind mirror $M_c$.

There are four beams of interest leaving the cavity. The main output beam is '$Out_b$' which goes to the interferometer and the stabilization of which is the main purpose of the modecleaner. There are two beams coming from $M_a$: the directly reflected input beam and the beam '$Out_a$' which is a fraction of the cavity eigenmode. The cavity is well aligned to the incoming beam (which we consider fixed), if these two beams are perfectly superimposed. By taking two quadrant diodes with two different lens systems and appropriately demodulating their outputs, we can obtain four independent error signals, similar to the case of a simple two-mirror Fabry-Perot cavity. The longitudinal locking signal is also obtained from these two interfering beams (using the Pound-Drever-Hall scheme).

[1] To avoid confusion, we call a mirror or other component **misoriented** in this section, if its angular orientation differs from its reference orientation. The resulting movement of beams (e.g. cavity eigenmodes) will be called **misalignment**. An interferometer is called **well-aligned** if there are neither misorientations nor misalignments, i.e. components as well as beams are in their reference positions.

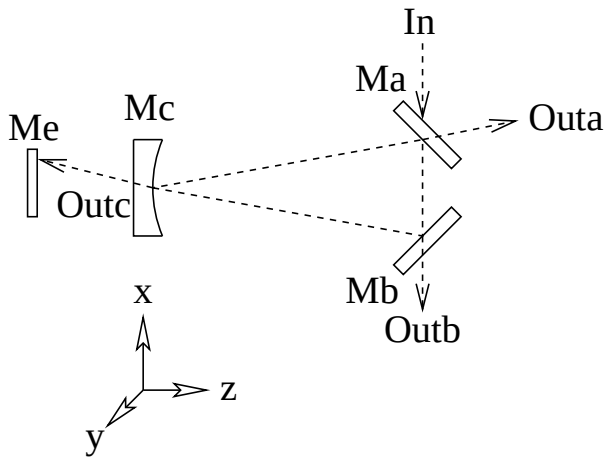



Figure 1: Schematic diagram of the TAMA modecleaner cavity seen from above.

In the three-mirror cavity, there are two additional degrees of freedom. They are linear combinations of movements of all three mirrors, as computed below. We will call them "neutral modes" because they have no influence on either the interference of the incoming beam with the cavity eigenmode nor on the outgoing beam. They can be used to control the spot position on the far mirror $M_c$. This is described in more detail below.

In the ray-tracing program, we first compute the well-aligned case (i.e., all mirrors are hit in their center) as reference. We call $P_a$, $P_b$ and $P_c$ the points where the axis of the eigenmode intersects the mirrors $M_a$, $M_b$ and $M_c$, respectively. After misorienting one particular mirror by the small angle $\varepsilon$, we recompute the eigenmode axis, compare it with the well-aligned case and divide the difference by $\varepsilon$. The main results are:

- The shifts of the spots $P_a$, $P_b$ and $P_c$.
- The angles $\gamma_a$, $\gamma_b$ and $\gamma_c$ between the beams '$Out_a$', '$Out_b$', '$Out_c$' and their respective references. For the vertical misalignments which are considered separately, we call these angles $\delta_a$, $\delta_b$ and $\delta_c$, respectively. We also compute the angle $\gamma_d$ ($\delta_d$) between the directly reflected incoming beam and its reference for the case that $M_a$ is misoriented.
- For '$Out_a$' in the case of misorienting $M_a$, we also compute the angle $\gamma'_a = \gamma_a - \gamma_d$, which is the angle between the beam leaving the cavity and the directly reflected beam, because this is the angle between the interfering wavefronts that is detected by the quadrant diode.
- As described above we finally compute the angle $\theta$ which describes the 'character' of the misalignment at the waist. It is given by $\theta^w = \arctan(\gamma'_a z_R / \Delta z_{waist})$.

The waist of the cavity eigenmode is located halfway between the mirrors $M_a$ and

$M_b$. Its Rayleigh range is given by

$$z_R = \sqrt{\frac{L_{RT}}{2}\left(R - \frac{L_{RT}}{2}\right)} = 7.126\,\mathrm{m}, \tag{1}$$

where $L_{RT}/2$ is one half of the round-trip distance:

$$L_{RT}/2 = \sqrt{L^2 + d^2/4} + d/2 = 9.8385\,\mathrm{m}. \tag{2}$$

# 3 Horizontal misalignments:

By 'horizontal' misalignments we mean that a mirror is rotated around the vertical axis, i.e. beam spots move horizontally (in the plane of the modecleaner cavity). Angles are counted as positive if a mirror is rotated *clockwise*, if seen from above (as in Figure 1).

We introduce the common- and differential mode motion of mirrors $M_a$ and $M_b$ by defining angles $\alpha_+$ and $\alpha_-$. Furthermore we introduce the "neutral" mode $\alpha_n$ as follows:

| | $M_a$ | $M_b$ | $M_c$ |
|---|---|---|---|
| $\alpha_+$ | $\alpha_a = \alpha_+$ | $\alpha_b = \alpha_+$ | $\alpha_c = 0$ |
| $\alpha_-$ | $\alpha_a = \alpha_-$ | $\alpha_b = -\alpha_-$ | $\alpha_c = 0$ |
| $\alpha_n$ | $\alpha_a = \alpha_n$ | $\alpha_b = \alpha_n$ | $\alpha_c = 0.701\alpha_n$ |

(3)

$$\alpha_+ = \frac{\alpha_a + \alpha_b}{2}, \tag{4}$$

$$\alpha_- = \frac{\alpha_a - \alpha_b}{2}, \tag{5}$$

The results of the raytracing program are given in Table 1.

| Cause | $P_a$ $\Delta x$ | $P_a$ $\Delta z$ | waist $\Delta z$ | $Out_a$ $\gamma_a$ | $Out'_a$ $\gamma'_a = \gamma_a - \gamma_d$ | $\theta^w$ |
|---|---|---|---|---|---|---|
| $\alpha_a$ | $-9.740\,\mathrm{m}\cdot\alpha_a$ | $9.840\,\mathrm{m}\cdot\alpha_a$ | $9.739\,\mathrm{m}\cdot\alpha_a$ | $0.981\,\alpha_a$ | $-1.019\,\alpha_a$ | $-36.72°$ |
| $\alpha_b$ | $9.538\,\mathrm{m}\cdot\alpha_b$ | $-9.636\,\mathrm{m}\cdot\alpha_b$ | $-9.739\,\mathrm{m}\cdot\alpha_b$ | $-1.019\,\alpha_b$ | $-1.019\,\alpha_b$ | $36.72°$ |
| $\alpha_c$ | $0.288\,\mathrm{m}\cdot\alpha_c$ | $-0.291\,\mathrm{m}\cdot\alpha_c$ | $0.000\,\mathrm{m}\cdot\alpha_c$ | $2.906\,\alpha_c$ | $2.906\,\alpha_c$ | $-90.00°$ |
| $\alpha_+$ | $-0.202\,\mathrm{m}\cdot\alpha_+$ | $0.204\,\mathrm{m}\cdot\alpha_+$ | $0.000\,\mathrm{m}\cdot\alpha_+$ | $-0.039\,\alpha_+$ | $-2.039\,\alpha_+$ | $90.00°$ |
| $\alpha_-$ | $-19.278\,\mathrm{m}\cdot\alpha_-$ | $19.477\,\mathrm{m}\cdot\alpha_-$ | $19.477\,\mathrm{m}\cdot\alpha_-$ | $2.000\,\alpha_-$ | $0.000\,\alpha_-$ | $0.00°$ |
| $\alpha_n$ | $0.000\,\mathrm{m}\cdot\alpha_n$ | $0.000\,\mathrm{m}\cdot\alpha_n$ | $0.000\,\mathrm{m}\cdot\alpha_n$ | $2.000\,\alpha_n$ | $0.000\,\alpha_n$ | — |

| Cause | $P_b$ $\Delta x$ | $P_b$ $\Delta z$ | $Out_b$ $\gamma_b$ | $P_c$ $\Delta x$ | $Out_c$ $\gamma_c$ | d. refl. $\gamma_d$ | End $\Delta x$ |
|---|---|---|---|---|---|---|---|
| $\alpha_a$ | $9.538\,\mathrm{m}\cdot\alpha_a$ | $9.637\,\mathrm{m}\cdot\alpha_a$ | $1.019\,\alpha_a$ | $-0.291\,\mathrm{m}\cdot\alpha_a$ | $-1.019\,\alpha_a$ | $2\cdot\alpha_a$ | $-2.411\,\mathrm{m}\cdot\alpha_a$ |
| $\alpha_b$ | $-9.740\,\mathrm{m}\cdot\alpha_b$ | $-9.841\,\mathrm{m}\cdot\alpha_b$ | $1.019\,\alpha_b$ | $-0.291\,\mathrm{m}\cdot\alpha_b$ | $0.981\,\alpha_b$ | $0\cdot\alpha_b$ | $1.749\,\mathrm{m}\cdot\alpha_b$ |
| $\alpha_c$ | $0.288\,\mathrm{m}\cdot\alpha_c$ | $0.291\,\mathrm{m}\cdot\alpha_c$ | $-2.906\,\alpha_c$ | $28.596\,\mathrm{m}\cdot\alpha_c$ | $2.906\,\alpha_c$ | $0\cdot\alpha_c$ | $34.642\,\mathrm{m}\cdot\alpha_c$ |
| $\alpha_+$ | $-0.202\,\mathrm{m}\cdot\alpha_+$ | $-0.204\,\mathrm{m}\cdot\alpha_+$ | $2.039\,\alpha_+$ | $-0.581\,\mathrm{m}\cdot\alpha_+$ | $-0.039\,\alpha_+$ | $2\cdot\alpha_+$ | $-0.662\,\mathrm{m}\cdot\alpha_+$ |
| $\alpha_-$ | $19.278\,\mathrm{m}\cdot\alpha_-$ | $19.477\,\mathrm{m}\cdot\alpha_-$ | $0.000\,\alpha_-$ | $0.000\,\mathrm{m}\cdot\alpha_-$ | $-2.000\,\alpha_-$ | $2\cdot\alpha_-$ | $-4.160\,\mathrm{m}\cdot\alpha_-$ |
| $\alpha_n$ | $0.000\,\mathrm{m}\cdot\alpha_n$ | $0.000\,\mathrm{m}\cdot\alpha_n$ | $0.000\,\alpha_n$ | $19.478\,\mathrm{m}\cdot\alpha_n$ | $2.000\,\alpha_n$ | $2\cdot\alpha_n$ | $23.639\,\mathrm{m}\cdot\alpha_n$ |

Table 1: Results of the ray-tracing program for horizontal misalignments of the TAMA modecleaner.

# 4 Vertical misalignments:

In the modecleaners, the horizontal and vertical axes are *not* similar. The results of the raytracing program for vertical misalignments are given in Table 2. Note, for example, that a small vertical tilt $\beta_{\rm a}$ of the input mirror $\rm M_a$ (which is hit under approximately 45° from the incoming beam) causes a deflection of the reflected beam by only $\delta_{\rm d} = 1.43\beta_{\rm a}$ as compared to $\gamma_{\rm d} = 2\alpha_{\rm a}$ in the horizontal case. Another example is the tilt of $\rm M_c$ which, if horizontal, causes a pure angular misalignment at the waist. A vertical tilt of $\rm M_c$, on the other hand, shifts the cavity eigenmode downwards parallely, without changing any angles. Angles are now counted as positive when the mirror normal moves downward from the reference direction.

As before we introduce the common- and differential mode motion of mirrors $\rm M_a$ and $\rm M_b$ by defining angles $\beta_+$ and $\beta_-$. Furthermore we introduce the "neutral" mode $\beta_{\rm n}$ as follows:

| | $\rm M_a$ | $\rm M_b$ | $\rm M_c$ |
|---|---|---|---|
| $\beta_+$ | $\beta_{\rm a} = \beta_+$ | $\beta_{\rm b} = \beta_+$ | $\beta_{\rm c} = 0$ |
| $\beta_-$ | $\beta_{\rm a} = \beta_-$ | $\beta_{\rm b} = -\beta_-$ | $\beta_{\rm c} = 0$ |
| $\beta_{\rm n}$ | $\beta_{\rm a} = \beta_{\rm n}$ | $\beta_{\rm b} = \beta_{\rm n}$ | $\beta_{\rm c} = -0.4986\beta_{\rm n}$ |

(6)

$$\beta_+ = \frac{\beta_{\rm a} + \beta_{\rm b}}{2}, \tag{7}$$

$$\beta_- = \frac{\beta_{\rm a} - \beta_{\rm b}}{2}, \tag{8}$$

The results of the raytracing program are given in Table 2.

# 5 Degrees of freedom

The most important alignment task is to superimpose the axis of the cavity eigenmode with the axis of the incoming beam. This requires the control of four degrees of freedom. For this purpose, in the differential wavefront sensing method, we place two quadrant detectors with different lens systems in the beam reflected from $\rm M_a$. The interference between the directly reflected incoming beam, which is phase modulated at an RF frequency, and the beam '$\rm Out_a$' leaking out of the cavity contains enough information to lock the cavity longitudinally and to obtain alignment error signals for those four degrees of freedom that determine the superposition of the incoming beam and the cavity eigenmode.

In particular, we now assume all mirrors to be slightly misoriented and compute the combined signals which are obtained by demodulating the outputs of two quadrant detectors, one (called $X_I$) with $\Phi = 0°$ and the other one (called $X_Q$) with $\Phi = 90°$ of extra phase shift. We scale parallel shifts $\Delta y$ or $\Delta z$ with the appropriate factor $z_{\rm R}$ and obtain for horizontal misalignments:

$$X_I = -1.019\,\alpha_{\rm a} - 1.019\,\alpha_{\rm b} + 2.906\,\alpha_{\rm c} = -1.019\,\alpha_+ + 2.906\,\alpha_{\rm c}, \tag{9}$$

$$X_Q = 1.366\,\alpha_{\rm a} - 1.366\,\alpha_{\rm b} = 1.366\,\alpha_-, \tag{10}$$

| Cause | $P_a$ $\Delta y$ | waist $\Delta y$ | $Out_a$ $\delta_a$ | $Out'_a$ $\delta'_a = \delta_a - \delta_d$ | $\theta^w$ |
|---|---|---|---|---|---|
| $\beta_a$ | $-3.670\,\mathrm{m} \cdot \beta_a$ | $-3.740\,\mathrm{m} \cdot \beta_a$ | $-0.718\,\beta_a$ | $0.704\,\beta_a$ | $-53.27°$ |
| $\beta_b$ | $-3.810\,\mathrm{m} \cdot \beta_b$ | $-3.740\,\mathrm{m} \cdot \beta_b$ | $-0.704\,\beta_b$ | $-0.704\,\beta_b$ | $53.27°$ |
| $\beta_c$ | $-15.000\,\mathrm{m} \cdot \beta_c$ | $-15.000\,\mathrm{m} \cdot \beta_c$ | $0.000\,\beta_c$ | $0.000\,\beta_c$ | $0.0°$ |
| $\beta_+$ | $-7.480\,\mathrm{m} \cdot \beta_+$ | $-7.480\,\mathrm{m} \cdot \beta_+$ | $-1.421\,\beta_+$ | $0.000\,\beta_+$ | $0.0°$ |
| $\beta_-$ | $0.141\,\mathrm{m} \cdot \beta_-$ | $0.000\,\mathrm{m} \cdot \beta_-$ | $-0.014\,\beta_-$ | $1.407\,\beta_-$ | $90.0°$ |
| $\beta_n$ | $0.000\,\mathrm{m} \cdot \beta_n$ | $0.000\,\mathrm{m} \cdot \beta_n$ | $-1.421\,\beta_n$ | $0.000\,\beta_n$ | — |

| Cause | $P_b$ $\Delta y$ | $Out_b$ $\delta_b$ | $P_c$ $\Delta y$ | $Out_c$ $\delta_c$ | d. refl. $\delta_d$ | End $\Delta y$ |
|---|---|---|---|---|---|---|
| $\beta_a$ | $-3.810\,\mathrm{m} \cdot \beta_a$ | $0.704\,\beta_a$ | $-10.661\,\mathrm{m} \cdot \beta_a$ | $0.704\,\beta_a$ | $1.42 \cdot \beta_a$ | $-12.125\,\mathrm{m} \cdot \beta_a$ |
| $\beta_b$ | $-3.670\,\mathrm{m} \cdot \beta_b$ | $-0.704\,\beta_b$ | $-10.661\,\mathrm{m} \cdot \beta_b$ | $0.718\,\beta_b$ | $0.00 \cdot \beta_b$ | $-12.155\,\mathrm{m} \cdot \beta_b$ |
| $\beta_c$ | $-15.000\,\mathrm{m} \cdot \beta_c$ | $0.000\,\beta_c$ | $-15.000\,\mathrm{m} \cdot \beta_c$ | $0.000\,\beta_c$ | $0 \cdot \beta_c$ | $-15.000\,\mathrm{m} \cdot \beta_c$ |
| $\beta_+$ | $-7.480\,\mathrm{m} \cdot \beta_+$ | $0.000\,\beta_+$ | $-21.323\,\mathrm{m} \cdot \beta_+$ | $1.421\,\beta_+$ | $-1.421 \cdot \beta_+$ | $-24.280\,\mathrm{m} \cdot \beta_+$ |
| $\beta_-$ | $-0.141\,\mathrm{m} \cdot \beta_-$ | $1.407\,\beta_-$ | $0.000\,\mathrm{m} \cdot \beta_-$ | $-0.014\,\beta_-$ | $-1.421 \cdot \beta_-$ | $0.030\,\mathrm{m} \cdot \beta_-$ |
| $\beta_n$ | $0.000\,\mathrm{m} \cdot \beta_n$ | $0.000\,\beta_n$ | $-13.843\,\mathrm{m} \cdot \beta_n$ | $1.421\,\beta_n$ | $-1.420 \cdot \beta_n$ | $-16.800\,\mathrm{m} \cdot \beta_n$ |

Table 2: Results of the ray-tracing program for vertical misalignments of the TAMA modecleaner.

and for vertical misalignments:

$$Y_I = 0.7035\,\beta_a - 0.7035\,\beta_b = 0.7035\,\beta_-, \tag{11}$$

$$Y_Q = -0.524\,\beta_a - 0.524\,\beta_b - 2.105\,\beta_c = -0.524\,\beta_+ - 2.105\,\beta_c. \tag{12}$$

Using simple linear algebra we also find the "neutral" modes from these results. They are given in tables (3) and (6) above.

By inspecting tables 1 and 2 one finds that the main effect of these "neutral modes" is to shift the spot position on the far end mirror $M_c$. Hence their control is not absolutely essential for the basic function of the modecleaner. In the present situation, there is neither feedback to $M_c$ nor is the spot position on $M_c$ monitored. One problem with this approach might be that according to equations (9) to (12), a small motion of $M_c$ translates into error signals at the reflected light port which is $3 \ldots 4$ times larger than the error signal for a motion of the front mirrors of comparable amplitued. This huge error signal is then fed back the the *front* mirrors which are forced to compensate $M_c$'s misorientation although they are not very efficient actuators for that purpose. Furthermore, such feedback shifts the spot position on $M_c$ which due to the curved mirror surface again causes the whole circle to start. It is possible that some of the present dynamic range problems may be relieved by sensing and feeding back all degrees of freedom.

Because the main effect of the "neutral modes" is to shift the spot position on the far end mirror $M_c$, the most straightforward approach is to sense this spot position (with a DC quadrant detector looking at the transmitted light). In tables 1 and 2 above, this shift is computed as $\Delta x$ or $\Delta y$ of point Pc, respectively.

However, the detector is located at a distance $e = 2.08\,\mathrm{m}$ behind $M_c$, and the spot

position detected is hence a linear combination[2] of $\Delta x$ or $\Delta y$ of point Pc, respectively with the angle $\delta_\mathrm{c}$.

This shift of the beam spot on the photodetector is also calculated by the program and printed as "End $\Delta y$" in Tables 1 and 2.

If we also scale it by the factor $z_\mathrm{R}$ for consistency (although the way of detection is different and hence an arbitrary scale factor appears anyway), and call the error signals thus obtained $X_E$ and $Y_E$, we get

$$X_E = -0.3383\,\alpha_\mathrm{a} + 0.2455\,\alpha_\mathrm{b} + 4.8613\,\alpha_\mathrm{c} \tag{13}$$
$$= -0.0464\,\alpha_+ - 0.2919\,\alpha_- + 4.8613\,\alpha_\mathrm{c}, \tag{14}$$
$$Y_E = -1.701\,\beta_\mathrm{a} - 1.705\,\beta_\mathrm{b} - 2.105\,\beta_\mathrm{c} \tag{15}$$
$$= -1.703\,\beta_+ + 0.002\,\beta_- - 2.105\,\beta_\mathrm{c}. \tag{16}$$

We see that for the horizontal direction, the additional error signal is dominated by the misorientation of $\mathrm{M_c}$ and could directly be fed back to that mirror, while for the vertical direction the signals are more mixed. However, they are still sufficiently linearly independent such that by inversion of the well-conditioned matrix error signals for all three mirrors can be found. This matrix inversion yields

$$\begin{array}{rcrrr} \alpha_\mathrm{a} &=& -1.6726\,X_I & +1.4273\,X_Q & +X_E \\ \alpha_\mathrm{b} &=& -1.6726\,X_I & -X_Q & +X_E \\ \alpha_\mathrm{c} &=& -0.0319\,X_I & +0.1498\,X_Q & +0.7015\,X_E \end{array} \tag{17}$$

$$\begin{array}{rcrrr} \beta_\mathrm{a} &=& +1.6785\,Y_I & +Y_Q & -Y_E \\ \beta_\mathrm{b} &=& -1.6785\,Y_I & +Y_Q & -Y_E \\ \beta_\mathrm{c} &=& -0.0015\,Y_I & -1.6186\,Y_Q & +0.4987\,Y_E \end{array} \tag{18}$$

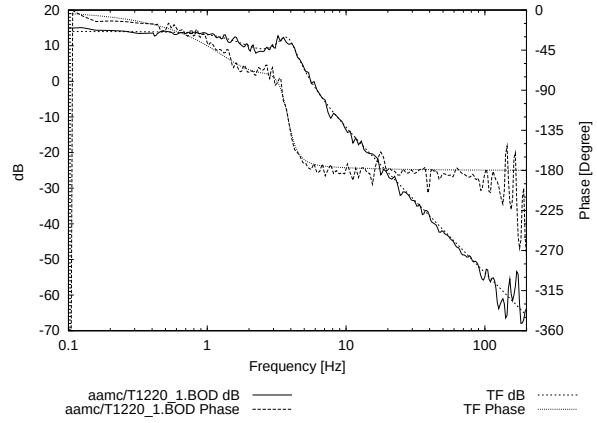
where scaling factors of 1/0.301462 and 1/0.42418, respectively, have been applied to make the most common coefficients unity. Note that the present alignment system uses only a $2 \times 2$ matrix without $X_E$, $Y_E$, $\alpha_\mathrm{c}$ and $\beta_\mathrm{c}$.

# 6 Actuator transfer functions

The transfer functions from all actuators to all sensors have been measured by K. Arai. The results were fitted with LISO and are shown on the following pages.

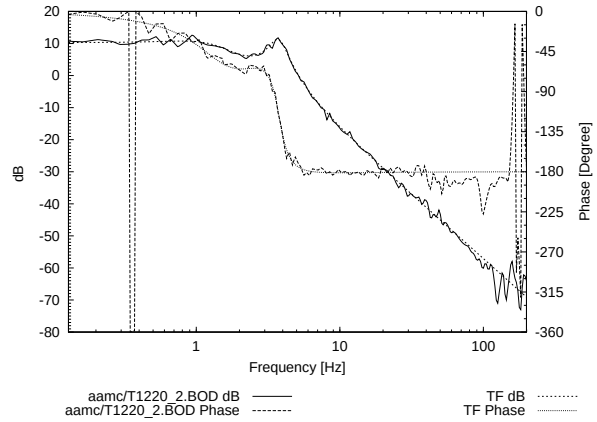
[2] Many thanks to K. Arai for noting this point.

## 6.1 Input Mirror Yaw coil ("m1xcoil")



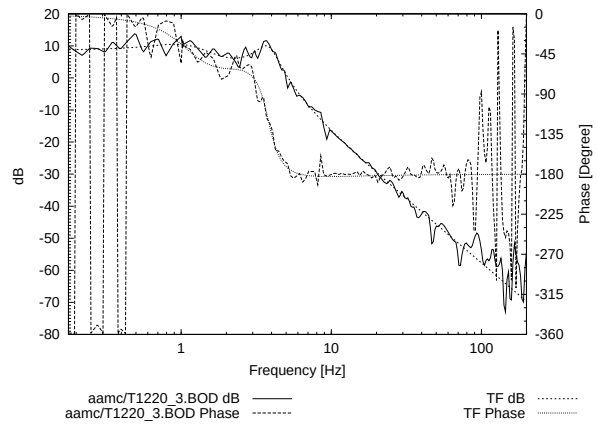

Pi Yaw error T1220_1.BOD

```
pole0:f = 1.6798911 +- 75.2m (4.48%)
pole0:q = 636.47668m +- 21.6m (3.39%)
zero0:f = 3.1323094 +- 124.6m (3.98%)
zero0:q = 969.43027m +- 64m (6.6%)
pole1:f = 3.7858238 +- 25.9m (0.684%)
pole1:q = 3.3333594 +- 177.3m (5.32%)
factor = 5.0043609 +- 40.31m (0.805%)
```




Pi/2 Yaw error T1220_2.BOD

```
pole0:f = 1.3598055 +- 43.77m (3.22%)
pole0:q = 830.24256m +- 19.97m (2.41%)
zero0:f = 2.4762582 +- 72.09m (2.91%)
zero0:q = 842.25894m +- 47.99m (5.7%)
pole1:f = 3.7674492 +- 19.53m (0.518%)
pole1:q = 3.711112 +- 128m (3.45%)
factor = 3.2644629 +- 29.84m (0.914%)
```



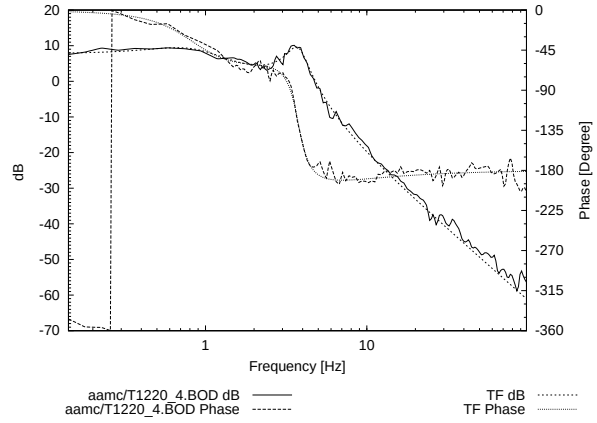

End QPD DC Yaw T1220_3.BOD

```
pole0:f = 1.1584155 +- 66.98m (5.78%)
pole0:q = 948.86053m +- 53.28m (5.61%)
zero0:f = 2.0163492 +- 120.4m (5.97%)
zero0:q = 642.40736m +- 78.67m (12.2%)
pole1:f = 3.8080309 +- 46.85m (1.23%)
pole1:q = 3.0342956 +- 189.4m (6.24%)
factor = 2.7204019 +- 63.32m (2.33%)
```

Weighted Averages:

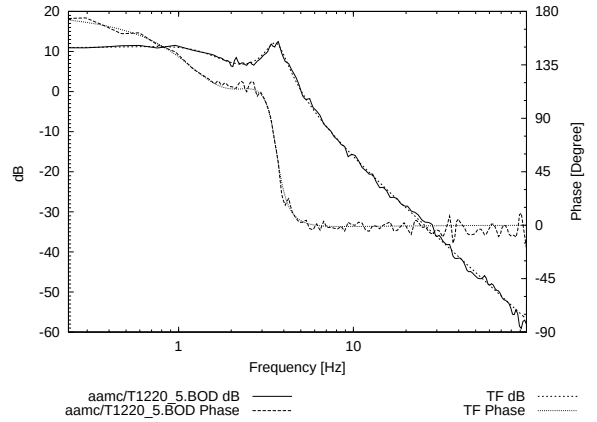
```
pole0:f = 1.372388
pole0:q = 0.755353
pole1:f = 3.777486
pole1:q = 3.455169
zero0:f = 2.508392
zero0:q = 0.840509
```

## 6.2 Output Mirror Yaw coil ("m2xcoil")



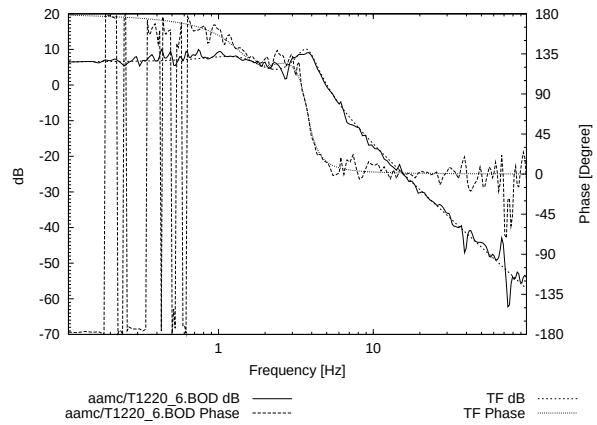

Pi Yaw error T1220_4.BOD

```
pole0:f = 864.55282m +- 31.63m (3.66%)
pole0:q = 801.30084m +- 22.58m (2.82%)
zero0:f = 1.7498418 +- 67.69m (3.87%)
zero0:q = 436.75937m +- 27.63m (6.33%)
pole1:f = 3.7556255 +- 19.05m (0.507%)
pole1:q = 3.6996359 +- 118m (3.19%)
factor = 2.4554303 +- 33.68m (1.37%)
```




Pi/2 Yaw error T1220_5.BOD

```
pole0:f = 1.3974404 +- 26.7m (1.91%)
pole0:q = 792.95797m +- 11.9m (1.5%)
zero0:f = 2.5625299 +- 44.11m (1.72%)
zero0:q = 877.60669m +- 27.81m (3.17%)
pole1:f = 3.630812 +- 9.79m (0.27%)
pole1:q = 3.9492515 +- 79.85m (2.02%)
factor = -3.5256953 +- -21.65m (0.614%)
```



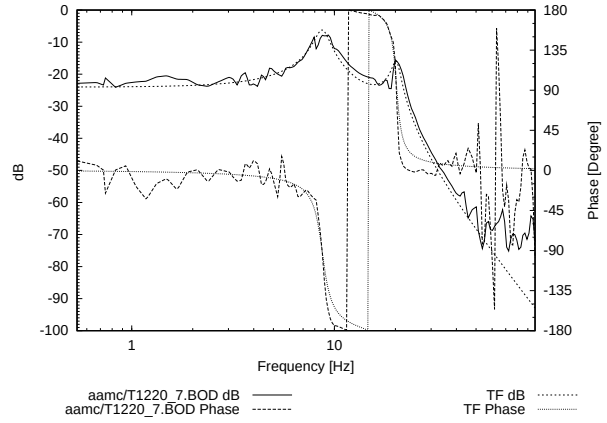

End QPD DC Yaw T1220_6.BOD

```
pole0:f = 1.5309455 +- 68.42m (4.47%)
pole0:q = 1.0838276 +- 56.89m (5.25%)
zero0:f = 2.2872224 +- 93.83m (4.1%)
zero0:q = 1.0726809 +- 108.7m (10.1%)
pole1:f = 3.7250662 +- 31.92m (0.857%)
pole1:q = 3.6033149 +- 192.1m (5.33%)
factor = -2.1236194 +- -32.95m (1.55%)
```

Weighted Averages:

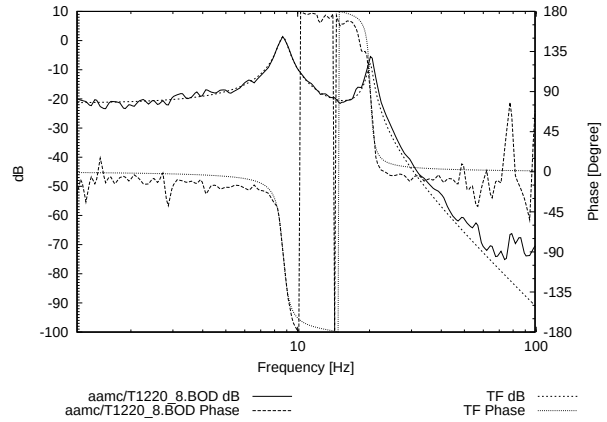
```
pole0:f = 1.204886
pole0:q = 0.804331
pole1:f = 3.661688
pole1:q = 3.842627
zero0:f = 2.316205
zero0:q = 0.669030
```

## 6.3 Input Mirror Pitch PZT ("m1ypzt")



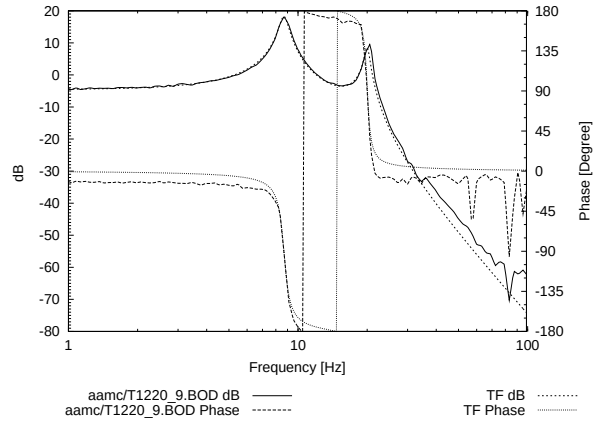

Pi Pitch error T1220_7.BOD

```
pole0:f = 8.721352 +- 31.18m (0.357%)
pole0:q = 6.3320232 +- 330.2m (5.22%)
pole1:f = 20.249388 +- 100.9m (0.498%)
pole1:q = 11.065166 +- 1.296 (11.7%)
factor = 62.862325m +- 1.561m (2.48%)
```




Pi/2 Pitch error T1220_8.BOD

```
pole0:f = 8.6836151 +- 12.55m (0.145%)
pole0:q = 11.122052 +- 405.8m (3.65%)
pole1:f = 20.153226 +- 42.54m (0.211%)
pole1:q = 20.638458 +- 1.887 (9.15%)
factor = 84.273794m +- 1.359m (1.61%)
```



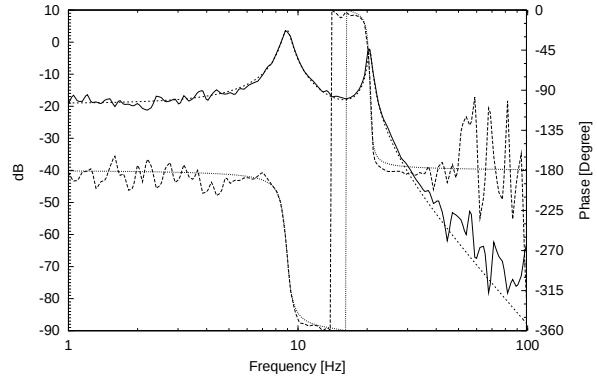

End QPD DC Pitch T1220_9.BOD

```
pole0:f = 8.6874272 +- 10.29m (0.118%)
pole0:q = 11.243448 +- 337.4m (3%)
pole1:f = 20.031176 +- 32.46m (0.162%)
pole1:q = 19.667483 +- 1.306 (6.64%)
factor = 590.1168m +- 7.742m (1.31%)
```

Weighted Averages:

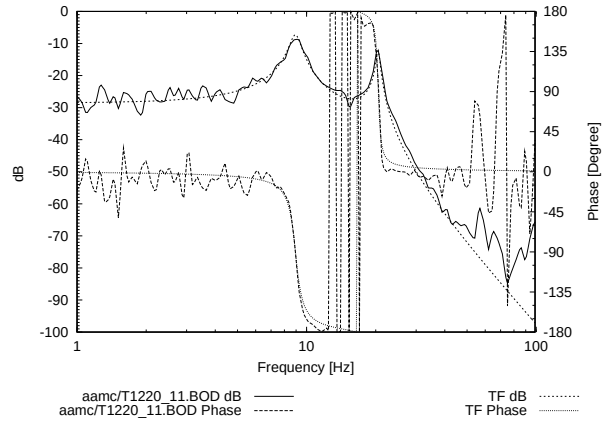
```
pole0:f = 8.688076
pole0:q = 9.340454
pole1:f = 20.08673
pole1:q = 16.34634
```

## 6.4 Output Mirror Pitch PZT ("m2ypzt")



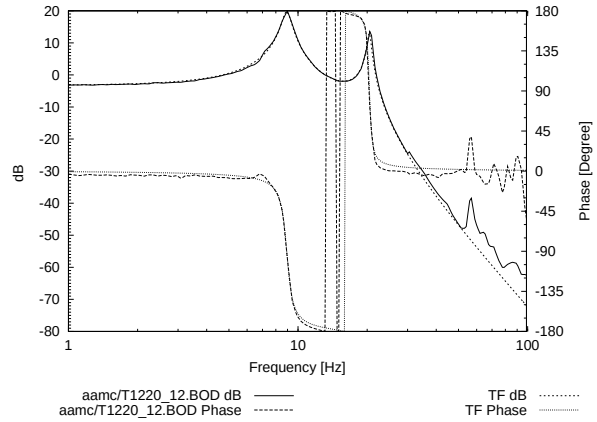

```
Pi Pitch error       T1220_10.BOD

pole0:f = 8.9312815 +- 8.936m (0.1%)
pole0:q = 11.488715 +- 297.2m (2.59%)
pole1:f = 20.561491 +- 19.32m (0.094%)
pole1:q = 29.210305 +- 1.681 (5.76%)
factor = -110.87858m +- -1.238m (1.12%)
```




```
Pi/2 Pitch error     T1220_11.BOD

pole0:f = 8.947484 +- 15.86m (0.177%)
pole0:q = 9.3417483 +- 354.6m (3.8%)
pole1:f = 20.507774 +- 30.04m (0.146%)
pole1:q = 27.878288 +- 2.424 (8.69%)
factor = 37.419176m +- 645.2u (1.72%)
```



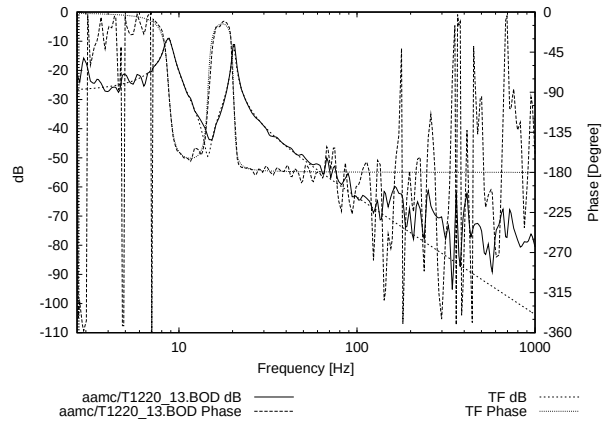

```
End QPD DC Pitch     T1220_12.BOD

pole0:f = 8.9347062 +- 5.266m (0.0589%)
pole0:q = 11.469713 +- 173.4m (1.51%)
pole1:f = 20.448379 +- 11.75m (0.0575%)
pole1:q = 27.958598 +- 926.9m (3.32%)
factor = 688.79131m +- 4.501m (0.653%)
```

Weighted Averages:

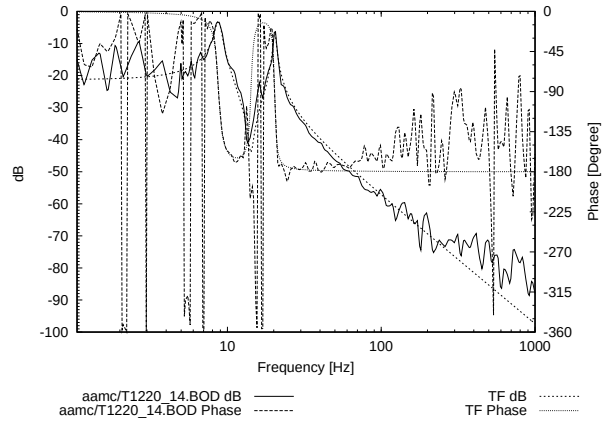
```
pole0:f = 8.934858
pole0:q = 11.15256
pole1:f = 20.48184
pole1:q = 28.21300
```

## 6.5 Input Mirror Pitch coil ("m1ycoil")



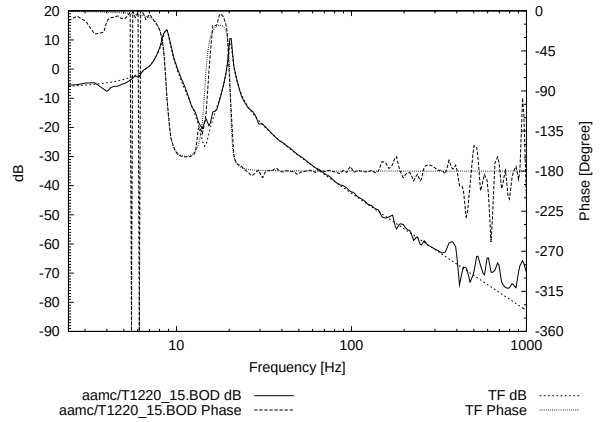

```
Pi Pitch error       T1220_13.BOD

pole0:f = 8.7707005 +- 19.38m (0.221%)
pole0:q = 10.585199 +- 541.9m (5.12%)
zero0:f = 14.532192 +- 125.6m (0.864%)
zero0:q = 15.263638 +- 4.742 (31.1%)
pole1:f = 20.291927 +- 33.79m (0.167%)
pole1:q = 27.515739 +- 2.803 (10.2%)
factor = 43.351314m +- 975.7u (2.25%)
```




```
Pi/2 Pitch error     T1220_14.BOD

pole0:f = 8.7581187 +- 23.51m (0.268%)
pole0:q = 10.768086 +- 656.8m (6.1%)
zero0:f = 14.220014 +- 291m (2.05%)
zero0:q =  16.341334 */ 1.856
pole1:f = 20.299641 +- 47.29m (0.233%)
pole1:q = 23.289386 +- 3.274 (14.1%)
factor = 86.457836m +- 3.091m (3.58%)
```



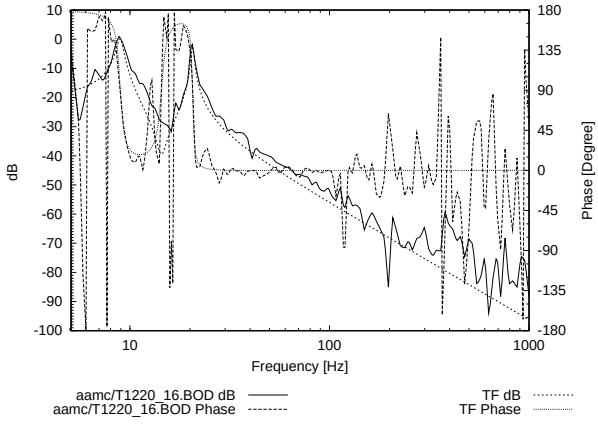

```
End QPD DC Pitch     T1220_15.BOD

pole0:f = 8.7555137 +- 8.536m (0.0975%)
pole0:q = 12.695277 +- 333.3m (2.63%)
zero0:f = 14.497233 +- 67.74m (0.467%)
zero0:q = 11.732884 +- 1.393 (11.9%)
pole1:f = 20.321918 +- 16.99m (0.0836%)
pole1:q = 28.889025 +- 1.515 (5.24%)
factor = 478.86074m +- 5.221m (1.09%)
```

Weighted Averages:

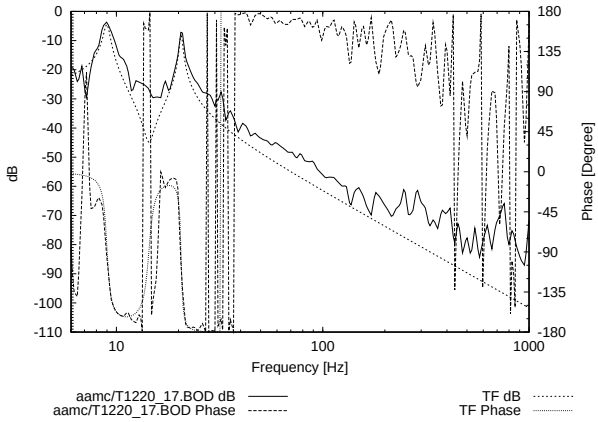
```
pole0:f = 8.757994
pole0:q = 11.90231
pole1:f = 20.31437
pole1:q = 27.83178
zero0:f = 14.49367
zero0:q = 12.04289
```

## 6.6 Output Mirror Pitch coil ("m2ycoil")



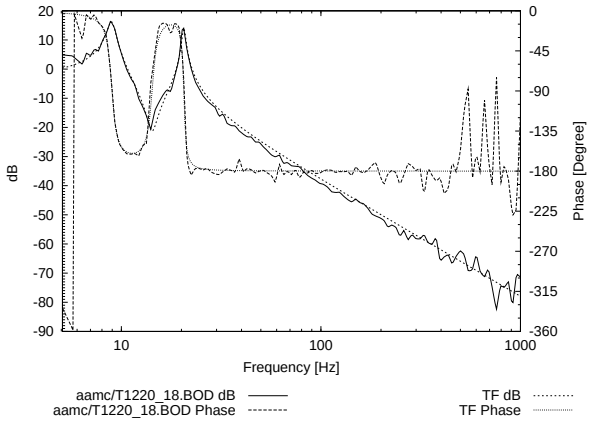

```
Pi Pitch error       T1220_16.BOD

pole0:f = 8.9343651 +- 27.53m (0.308%)
pole0:q = 14.281851 +- 1.225 (8.58%)
zero0:f = 14.399736 +- 235m (1.63%)
zero0:q = 10.829207 +- 272.2m (2.51%) > MAX
pole1:f = 20.60976 +- 44.38m (0.215%)
pole1:q = 33.383364 +- 5.186 (15.5%)
factor = -94.29963m +- -3.853m (4.09%)
```




```
Pi/2 Pitch error     T1220_17.BOD

pole0:f = 8.9970043 +- 32.66m (0.363%)
pole0:q = 15.935303 +- 1.807 (11.3%)
zero0:f = 14.374722 +- 317.3m (2.21%)
zero0:q = 10.838184 +- 241.2m (2.23%) > MAX
pole1:f = 20.577195 +- 61.19m (0.297%)
pole1:q = 33.687723 +- 6.336 (18.8%)
factor = 49.476234m +- 2.697m (5.45%)
```



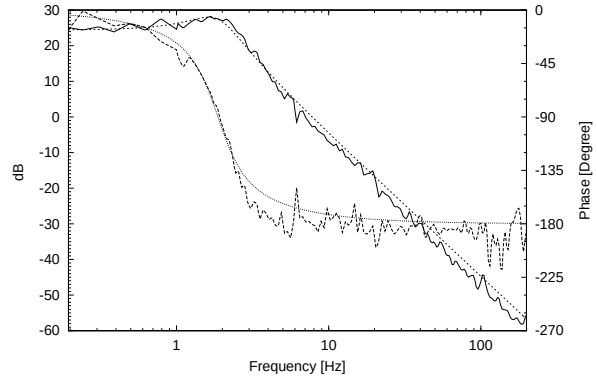

```
End QPD DC Pitch     T1220_18.BOD

pole0:f = 8.9471133 +- 10.88m (0.122%)
pole0:q = 11.299087 +- 344.5m (3.05%)
zero0:f = 14.280006 +- 68.37m (0.479%)
zero0:q = 10.79691 +- 1.226 (11.4%)
pole1:f = 20.579641 +- 17.91m (0.087%)
pole1:q = 26.245019 +- 1.311 (4.99%)
factor = 761.14975m +- 10.98m (1.44%)
```

Weighted Averages:

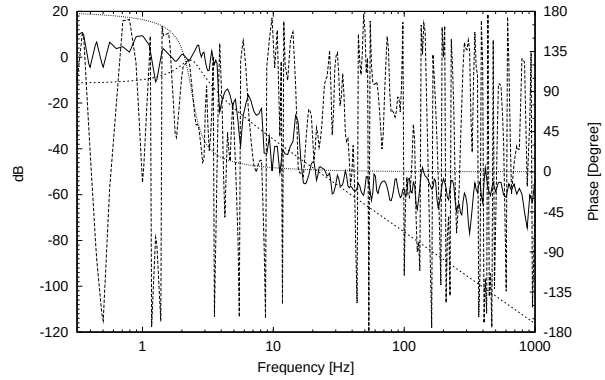
```
pole0:f = 8.949924
pole0:q = 11.66273
pole1:f = 20.58340
pole1:q = 26.94568
zero0:f = 14.29287
zero0:q = 10.79691
```

## 6.7 End Mirror Yaw Coil ("m3xcoil")



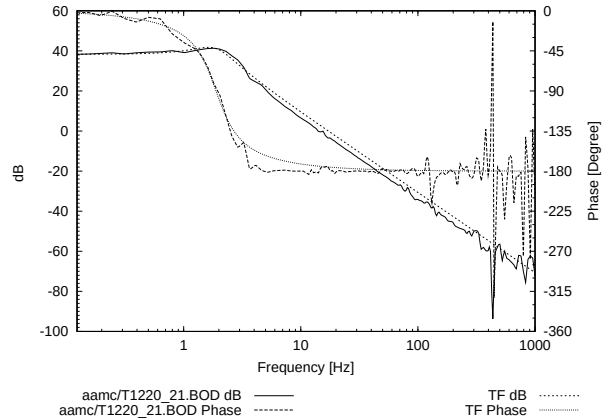

Pi Yaw error T1220_19.BOD

pole0:f = 1.8713374 +- 11.13m (0.595%)
pole0:q = 1.4166323 +- 29.77m (2.1%)
factor = 16.61447 +- 181.6m (1.09%)




Pi/2 Coil Yaw error T1220_20.BOD

pole0:f = 2.3769774 +- 87.34m (3.67%)
pole0:q = 3.1497205 +- 1.04 (33%)
factor = -270.8383m +- -49.8m (18.4%)



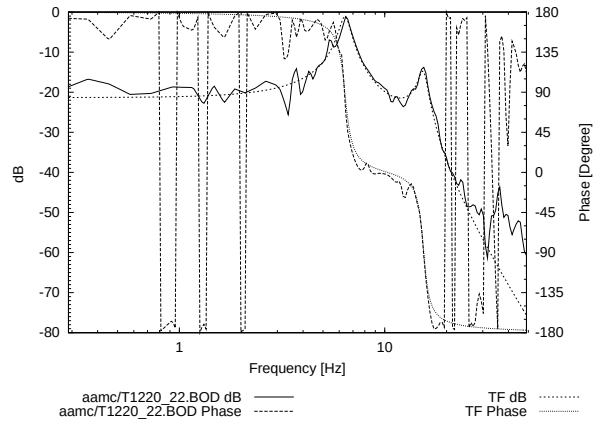

End QPD DC Yaw T1220_21.BOD

pole0:f = 1.8896638 +- 8.812m (0.466%)
pole0:q = 1.3891876 +- 22.04m (1.59%)
factor = 82.249734 +- 641.4m (0.78%)

Weighted Averages:

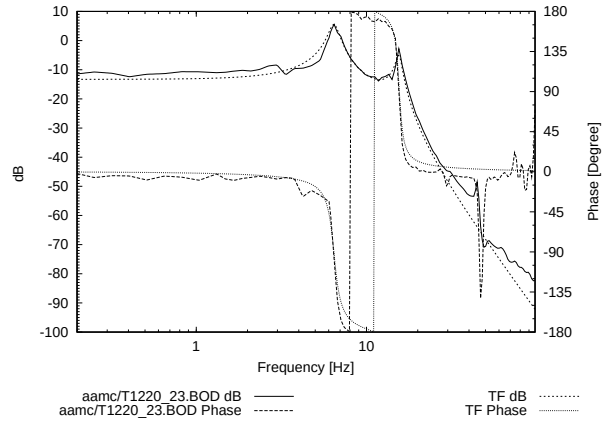
pole0:f = 1.8826161
pole0:q = 1.3989960

## 6.8 End Mirror Pitch PZT ("m3ypzt")



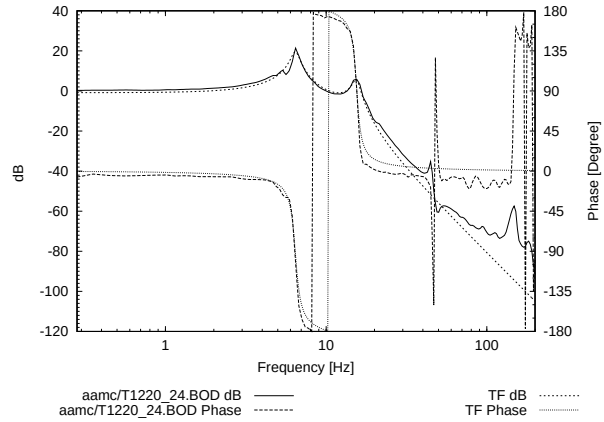

Pi Pitch error T1220_22.BOD

pole0:f = 6.4108286 +- 18.1m (0.282%)
pole0:q = 8.4989131 +- 459.5m (5.41%)
pole1:f = 15.438905 +- 94.27m (0.611%)
pole1:q = 10.134435 +- 1.324 (13.1%)
factor = -85.75237m +- -2.02m (2.36%)




Pi/2 Pitch error T1220_23.BOD

pole0:f = 6.4501997 +- 17.03m (0.264%)
pole0:q = 7.2412406 +- 310.6m (4.29%)
pole1:f = 15.548908 +- 63.98m (0.411%)
pole1:q = 12.259952 +- 1.299 (10.6%)
factor = 216.30985m +- 4.018m (1.86%)



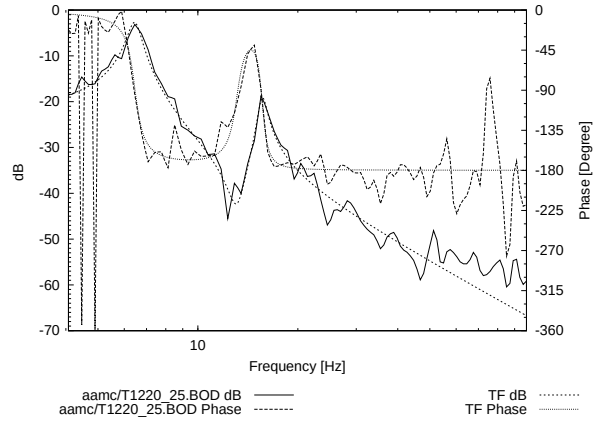

End QPD DC Pitch T1220_24.BOD

pole0:f = 6.441543 +- 11.96m (0.186%)
pole0:q = 8.6539705 +- 308.9m (3.57%)
pole1:f = 15.47822 +- 62.81m (0.406%)
pole1:q = 10.059454 +- 849.4m (8.44%)
factor = 904.141m +- 14m (1.55%)

Weighted Averages:

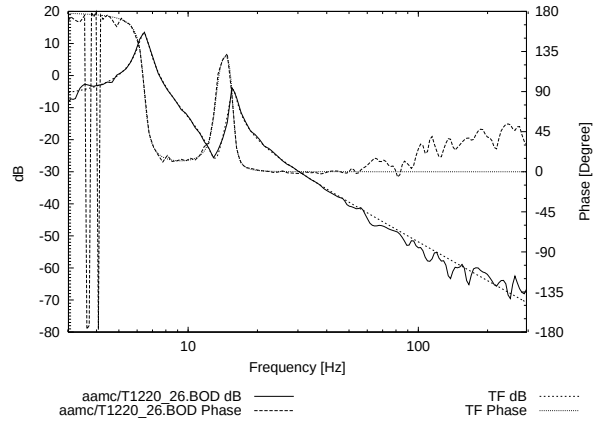
pole0:f = 6.436784
pole0:q = 8.052757
pole1:f = 15.49933
pole1:q = 10.58777

## 6.9 End Mirror Pitch coil ("m3ycoil")



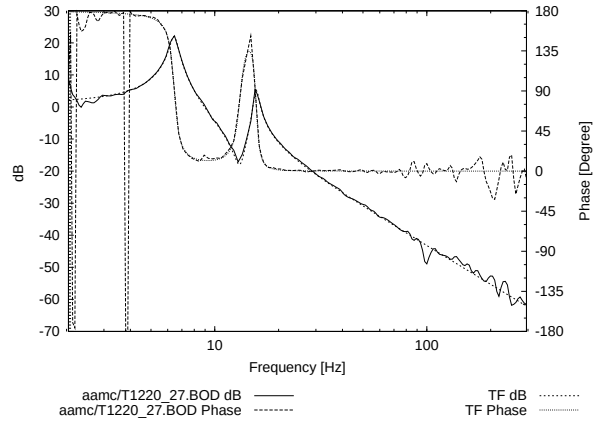

```
Pi Pitch error       T1220_25.BOD

pole0:f = 6.4562745 +- 12.24m (0.19%)
pole0:q = 10.880457 +- 519.8m (4.78%)
zero0:f = 13.118087 +- 141.4m (1.08%)
zero0:q = 10.177679 +- 2.379 (23.4%)
pole1:f = 15.621772 +- 66.15m (0.423%)
pole1:q = 16.829252 +- 2.755 (16.4%)
factor = 73.69469m +- 1.956m (2.65%)
```




```
Pi/2 Pitch error     T1220_26.BOD

pole0:f = 6.4394746 +- 4.254m (0.0661%)
pole0:q = 11.47595 +- 191.8m (1.67%)
zero0:f = 13.214751 +- 53.88m (0.408%)
zero0:q = 8.8553113 +- 623m (7.04%)
pole1:f = 15.652186 +- 26.16m (0.167%)
pole1:q = 15.990668 +- 937.6m (5.86%)
factor = -433.8639m +- -3.66m (0.843%)
```




```
End QPD DC Pitch     T1220_27.BOD

pole0:f = 6.4356973 +- 6.681m (0.104%)
pole0:q = 11.732925 +- 311.2m (2.65%)
zero0:f = 13.167754 +- 85.29m (0.648%)
zero0:q = 9.613809 +- 1.201 (12.5%)
pole1:f = 15.663831 +- 37.03m (0.236%)
pole1:q = 17.752179 +- 1.652 (9.31%)
factor = -1.1676377 +- -14.64m (1.25%)
```

Weighted Averages:

```
pole0:f = 6.439799
pole0:q = 11.48701
pole1:f = 15.65282
pole1:q = 16.45207
zero0:f = 13.19353
zero0:q = 9.075651
```

# 7 Calibration Factors

Both actuators and sensors have calibration factors which are not yet known. In order to find them (and test the above theroretical model), the LISO-fitted `factor`s from the transfer functions were fitted to simple model

$$\texttt{factor} = A_{\text{optical}} \times G_{\text{actuator}} \times G_{\text{sensor}}. \tag{19}$$

where *all* measurements were fitted simultaneously to one set of calibration constants $G_i$.

For that purpose a simple C program ("`prod.c`") was written which uses the Nelder-Mead Simplex algorithm to minimize

$$\chi^2(G_1, \ldots, G_m) = \sum_{i=1}^{n_{\text{data}}} w_i \left| f_i - \tilde{f}_i(G_1, \ldots, G_m) \right|^2 \tag{20}$$

where $f_i$ are the measured factors, $\tilde{f}_i$ their approximations from Equation (19), $w_i = 1/\sigma_i^2$ are the weights of each data point derived from their standard deviation (taken from the LISO fit), and $G_i$ the unknown calibration constants.

In fact the data set consists of two independent sets of data (for $x$ and $y$), but since this does not adversely affect the fit they were nevertheless fitted all in one set.

To remove an inherent ambiguity in the model (19), the calibration factor (sensitivity) for the end detector was arbitrarily fixed at unity:

$$\texttt{xe} = 1, \tag{21}$$

$$\texttt{ye} = 1. \tag{22}$$

Three measurements where the theoretically expected factor was zero (and the measured one was indeed small) were excluded from the fit. Hence there are 13 parameters and 24 data points.

After convergence of the Simplex algorithm, the solution was refined by a simplified Levenberg-Marquardt algorithm, and the standard errors of the parameters were computed (see Appendix of LISO manual for relevant formulae).

The results are:

| Parameter | est. | $\sigma$ | $\sigma$ |
|---|---|---|---|
| xi | 0.371443 | 0.0531476 | 14.308% |
| xq | −0.274189 | 0.0435769 | 15.893% |
| m1xcoil | −10.0941 | 1.56787 | 15.533% |
| m2xcoil | −8.55752 | 1.2643 | 14.774% |
| m3xcoil | 16.3377 | 1.83309 | 11.220% |
| yi | −0.313443 | 0.052152 | 16.638% |
| yq | 0.270632 | 0.0341316 | 12.612% |
| m1ypzt | −0.365368 | 0.0571409 | 15.639% |
| m2ypzt | −0.38773 | 0.0357978 | 9.233% |
| m3ypzt | −0.406073 | 0.077663 | 19.125% |
| m1ycoil | −0.25863 | 0.0416197 | 16.092% |
| m2ycoil | −0.43544 | 0.0878683 | 20.179% |
| m3ycoil | 0.66633 | 0.0813667 | 12.211% |

The following table shows the input data and residuals after the fit.

| $G_a$ | $G_s$ | $A_{\text{opt}}$ | $\sigma$ | meas. | calc. | ratio |
|---|---|---|---|---|---|---|
| m1xcoil | xi | −1.0190 | 0.04031 | 5.00436 | 3.82063 | -2.34 db |
| m1xcoil | xq | 1.3660 | 0.02984 | 3.26446 | 3.78069 | 1.28 db |
| m1xcoil | 1.0 | −0.3383 | 0.06332 | 2.72040 | 3.41485 | 1.97 db |
| m2xcoil | xi | −1.0190 | 0.03368 | 2.45543 | 3.23903 | 2.41 db |
| m2xcoil | xq | −1.3660 | 0.02165 | −3.52570 | −3.20516 | -0.83 db |
| m2xcoil | 1.0 | 0.2454 | 0.03295 | −2.12362 | −2.10002 | -0.10 db |
| m3xcoil | xi | 2.9060 | 0.1816 | 16.61447 | 17.63516 | 0.52 db |
| m3xcoil | 1.0 | 4.8613 | 0.6414 | 82.24973 | 79.42255 | -0.30 db |
| m1ypzt | yi | 0.7035 | 0.001561 | 0.06286 | 0.08057 | 2.16 db |
| m1ypzt | yq | −0.5240 | 0.001359 | 0.08427 | 0.05181 | -4.23 db |
| m1ypzt | 1.0 | −1.7010 | 0.007742 | 0.59012 | 0.62149 | 0.45 db |
| m2ypzt | yi | −0.7035 | 0.001238 | −0.11088 | −0.08550 | -2.26 db |
| m2ypzt | yq | −0.5240 | 0.0006452 | 0.03742 | 0.05498 | 3.34 db |
| m2ypzt | 1.0 | −1.7050 | 0.004501 | 0.68879 | 0.66108 | -0.36 db |
| m1ycoil | yi | 0.7035 | 0.0009757 | 0.04335 | 0.05703 | 2.38 db |
| m1ycoil | yq | −0.5240 | 0.003091 | 0.08646 | 0.03668 | -7.45 db |
| m1ycoil | 1.0 | −1.7010 | 0.005221 | 0.47886 | 0.43993 | -0.74 db |
| m2ycoil | yi | −0.7035 | 0.003853 | −0.09430 | −0.09602 | 0.16 db |
| m2ycoil | yq | −0.5240 | 0.002697 | 0.04948 | 0.06175 | 1.92 db |
| m2ycoil | 1.0 | −1.7050 | 0.01098 | 0.76115 | 0.74242 | -0.22 db |
| m3ypzt | yq | −2.1050 | 0.004018 | 0.21631 | 0.23133 | 0.58 db |
| m3ypzt | 1.0 | −2.1050 | 0.014 | 0.90414 | 0.85478 | -0.49 db |
| m3ycoil | yq | −2.1050 | 0.00366 | −0.43386 | −0.37960 | -1.16 db |
| m3ycoil | 1.0 | −2.1050 | 0.01464 | −1.16764 | −1.40263 | 1.59 db |

# 8 Port usage

We use **mode 7 - Advanced single chip mode**. In this mode, only internal RAM (4k) and Flash-ROM (128k) are used, and all I/O lines are available. In the other modes, many of them are used as address / data lines for external memory.

## 8.1 Port 1

8-bit I/O, can drive LED (sink 10 mA), TTL + 90 pF, Darlington.

**Registers** P1DDR - data direction register: 0=Input, 1=Output, default=0;
P1DR - data register, default=0.

**Usage** Output port for 8 status LED's, via HC04 Inverter. Logic is active Low, i.e. 0 = LED On, 1=LED Off

**P1.DR.BIT.B0** LED0 Ch1 Red

**P1.DR.BIT.B1** LED1 Ch2 Red

**P1.DR.BIT.B2** LED2 Ch3 Red

**P1.DR.BIT.B3** LED3 Ch4 Red

**P1.DR.BIT.B4** LED4 Ch5 Red

**P1.DR.BIT.B5** LED5 Ch6 Red

**P1.DR.BIT.B6** LED6 Green 1

**P1.DR.BIT.B7** LED7 Green 2

## 8.2 Port 2

8-bit I/O, can drive TTL + 90 pF, Darlington. Input: programmable pull-up ($50 \ldots 300\mu$A = $16 \ldots 100$k$\Omega$).

**Registers** P2DDR - data direction register: 0=Input, 1=Output, default=0;
P2DR - data register, default=0,
P2PCR - pull up register, 0=Off, 1=pull-up, default=0.

**Usage** Output port for 8 status LED's, via HC04 Inverter. Logic is active Low, i.e. 0 = LED On, 1=LED Off

**P2.DR.BIT.B0** LED8 Ch1 Yellow

**P2.DR.BIT.B1** LED9 Ch2 Yellow

**P2.DR.BIT.B2** LED10 Ch3 Yellow

**P2.DR.BIT.B3** LED11 Ch4 Yellow

**P2.DR.BIT.B4** LED12 Ch5 Yellow

**P2.DR.BIT.B5** LED13 Ch6 Yellow

**P2.DR.BIT.B6** LED14 unused

**P2.DR.BIT.B7** LED15 unused

## 8.3 Port 3

8-bit I/O, can drive TTL + 90 pF, Darlington.

**Registers** P3DDR - data direction register: 0=Input, 1=Output, default=0;
P3DR - data register, default=0.

**Usage:** Output port for LCD display

**P3.DR.BIT.B0** LCD Data bit 4

**P3.DR.BIT.B1** LCD Data bit 5

**P3.DR.BIT.B2** LCD Data bit 6

**P3.DR.BIT.B3** LCD Data bit 7

**P3.DR.BIT.B4** LCD R/S signal

**P3.DR.BIT.B5** LCD enable signal

**P3.DR.BIT.B6** unused

**P3.DR.BIT.B7** unused

## 8.4 Port 4

8-bit I/O, can drive TTL + 90 pF, Darlington. Input: programmable pull-up ($50 \ldots 300\mu$A = $16 \ldots 100$k$\Omega$).

**Registers** P4DDR - data direction register: 0=Input, 1=Output, default=0;
P4DR - data register, default=0,
P4PCR - pull up register, 0=Off, 1=pull-up, default=0.

**Usage** Digital Input, pull up enabled

**P4.DR.BIT.B0** via 74HC04 Inverter and Pullup - Digital Input 0

**P4.DR.BIT.B1** via 74HC04 Inverter and Pullup - Digital Input 1

**P4.DR.BIT.B2** via 74HC04 Inverter and Pullup - Digital Input 2

**P4.DR.BIT.B3** via 74HC04 Inverter and Pullup - Digital Input 3

**P4.DR.BIT.B4** via 74HC04 Inverter and Pullup - Digital Input 4

**P4.DR.BIT.B5** via 74HC04 Inverter and Pullup - Digital Input 5

**P4.DR.BIT.B6** - Digital Input 6

**P4.DR.BIT.B7** - Digital Input 7

**P4.DR.BIT.B0...P4.DR.BIT.B3** BCD switch 0...9

**P4.DR.BIT.B4** Reset key 1=pressed

**P4.DR.BIT.B5** left flipswitch 0=On

**P4.DR.BIT.B6** centre flipswitch 0=On

**P4.DR.BIT.B7** right flipswitch 0=On

## 8.5 Port 5

4-bit I/O, can drive LED (sink 10 mA), TTL + 90 pF, Darlington, Input: programmable pull-up ($50 \ldots 300\mu$A = $16 \ldots 100$k$\Omega$).

**Registers** P5DDR - data direction register: 0=Input, 1=Output, default=0;
P5DR - data register, default=0,
P5PCR - pull up register, 0=Off, 1=pull-up, default=0.

**Usage** DAC8420 Control (2 Chips 2 bits each)

**P5.DR.BIT.B0** DAC2 LD

**P5.DR.BIT.B1** DAC2 CS

**P5.DR.BIT.B2** DAC1 LD

**P5.DR.BIT.B3** DAC1 CS

## 8.6 Port 6

7-bit I/O, can drive TTL + 30 pF, Darlington,

**Registers** P6DDR - data direction register: 0=Input, 1=Output, default=0;
P6DR - data register, default=0.

**Usage** unused

**P6.DR.BIT.B0**

**P6.DR.BIT.B1**

**P6.DR.BIT.B2**

**P6.DR.BIT.B3**

**P6.DR.BIT.B4**

**P6.DR.BIT.B5**

**P6.DR.BIT.B6**

## 8.7 Port 7

8-bit Input / Analog Input / Analog Output.
8 channel 10-bit A/D on pins 0...7;
2 channel 8-bit D/A on pins 6 and 7.

**Registers** P7DR - data register, default=0.

**Usage** pin 0 – A/D group 0 : AD channel 0
pin 1 – A/D group 0 : AD channel 1
pin 2 – A/D group 0 : AD channel 2
pin 3 – A/D group 0 : AD channel 3
pin 4 – A/D group 1 : AD channel 4
pin 5 – A/D group 1 : AD channel 5
pin 6 – A/D group 1 : AD channel 6
pin 7 – A/D group 1 : AD channel 7

## 8.8 Port 8

5-bit I/O, also $IRQ_0$,...,$IRQ_3$.
pin 0 is I/O or $IRQ_0$, Schmitt-trigger input
pin 1 is I/O or $IRQ_1$, Schmitt-trigger input
pin 2 is I/O or $IRQ_2$, Schmitt-trigger input
pin 3 is I/O or $IRQ_3$,
pin 4 is I/O.
Output can drive TTL + 90 pF, Darlington.

**Registers** P8DDR - data direction register: 0=Input, 1=Output, default=0;
P8DR - data register, default=0.
IER - enables interrupt regardless of P8DDR.

**Usage** pin 0 – IRQ 0 – user pressbutton

## 8.9 Port 9

6-bit I/O, can drive TTL + 30 pF, Darlington,
also used as serial port and $IRQ_4$, $IRQ_5$.
pin 0 is I/O or $TxD_0$,
pin 1 is I/O or $TxD_1$,
pin 2 is I/O or $RxD_0$,
pin 3 is I/O or $RxD_1$,
pin 4 is I/O or $SCK_0$ or $IRQ_4$,
pin 5 is I/O or $SCK_1$ or $IRQ_5$.

**Registers** P9DDR - data direction register: 0=Input, 1=Output, default=0;
P9DR - data register, default=0.
serial mode / IRQ overrides P9DDR.

**Usage** pin 1 and pin 3 (SCI1) for serial communication with PC via RS232 buffer
pins 0,4 (SCI0) for synchronous serial data transfer to DAC8420

### 8.10 Port A

8-bit I/O, can drive TTL + 30 pF, Darlington,
Schmitt-trigger input
also integrated timer unit I/O, DMA output, timing pattern output.
pin 0 is I/O or $TP_0$ (out) or $TEND_0$ (out) or TCLKA (in),
pin 1 is I/O or $TP_1$ (out) or $TEND_1$ (out) or TCLKB (in),
pin 2 is I/O or $TP_2$ (out) or $TIOCA_0$ (I/O) or TCLKC (in),
pin 3 is I/O or $TP_3$ (out) or $TIOCB_0$ (I/O) or TCLKD (in),
pin 4 is I/O or $TP_4$ (out) or $TIOCA_1$ (I/O),
pin 5 is I/O or $TP_5$ (out) or $TIOCB_1$ (I/O),
pin 6 is I/O or $TP_6$ (out) or $TIOCA_2$ (I/O),
pin 7 is I/O or $TP_7$ (out) or $TIOCB_2$ (I/O).

**Registers** PADDR - data direction register: 0=Input, 1=Output, default=0;
PADR - data register, default=0.
if timing pattern output, PADDR bit must be set.

**Usage** pin 0 and pin 1 input from quadrature encoder switch
pin 2 implicit use to cascade connect timers 0 and 1 ("finetime")

### 8.11 Port B

8-bit I/O, can drive LED (sink 10 mA), TTL + 30 pF, Darlington,
pin 0...3 have Schmitt-trigger input.
also integrated timer unit I/O, DMA input, timing pattern output, A/D trigger.
pin 0 is I/O or $TP_8$ (out) or $TIOCA_3$ (I/O),
pin 1 is I/O or $TP_9$ (out) or $TIOCB_3$ (I/O),
pin 2 is I/O or $TP_{10}$ (out) or $TIOCA_4$ (I/O),
pin 3 is I/O or $TP_{11}$ (out) or $TIOCB_4$ (I/O),
pin 4 is I/O or $TP_{12}$ (out) or $TIOXA_4$ (out),
pin 5 is I/O or $TP_{13}$ (out) or $TIOXB_4$ (out),
pin 6 is I/O or $TP_{14}$ (out) or $DREQ_0$ (in),
pin 7 is I/O or $TP_{15}$ (out) or $DREQ_1$ (in), ADTRG.

**Registers** PBDDR - data direction register: 0=Input, 1=Output, default=0;
PBDR - data register, default=0.
if timing pattern output, PBDDR bit must be set.

**Usage** pin 0 (TIOCA3) has 1/2 sampling frequency as monitor.

# 9 Analog Inputs/Outputs

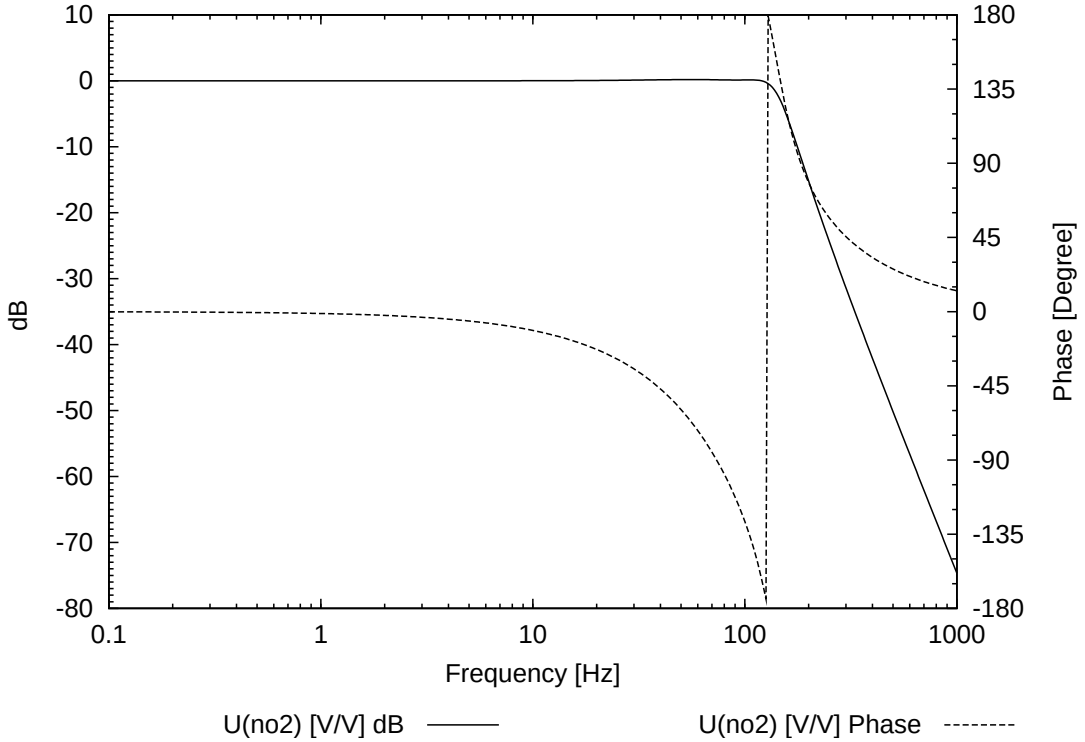
The A/D converter of the 3048 uses a stabilized +5 V reference. The nominal resolution is 10 bits, and the input range 0...5 Volt. By averaging a few samples, an effective resolution of 11...12 bits is achieved. All 8 analog input channels are used as follows:

| Channel No. | Preamp Gain | Input Range [V] | Signal |
|---|---|---|---|
| 0 | 5 | -0.667 ... +0.667 | I detector $x$ |
| 1 | 5 | -0.667 ... +0.667 | I detector $y$ |
| 2 | 5 | -0.667 ... +0.667 | Q detector $x$ |
| 3 | 5 | -0.667 ... +0.667 | Q detector $y$ |
| 4 | 1 | -3.33 ... +0.667 | End detector $x$ |
| 5 | 1 | -3.33 ... +0.667 | End detector $y$ |
| 6 | -1...-10 | 0...-0.5 to 0 ...-5 (var.) | Laser power |
| 7 | -1...-10 | 0...-0.5 to 0 ...-5 (var.) | Internal power |

All channels have analog 4-pole antialiasing filters with a 0.1 dB Tschebysheff characteristic and a corner frequency of 120 Hz. Their poles are

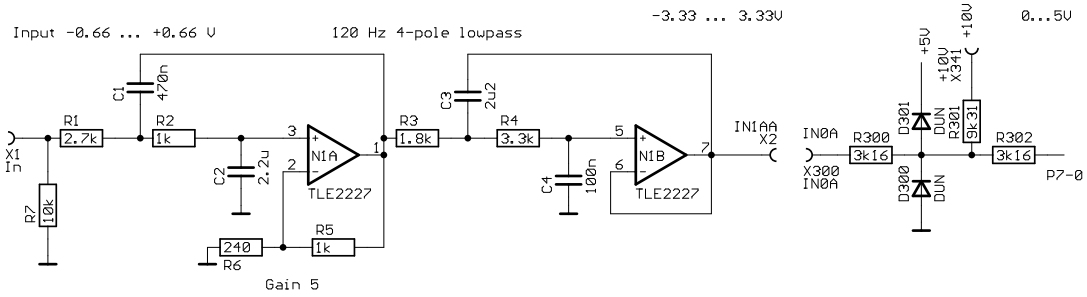
| Freq [Hz] | Q |
|---|---|
| $0.789 \times 120 = 94.68$ | 0.619 |
| $1.153 \times 120 = 138.36$ | 2.183 |

The following diagram shows the frequency response of the 120 Hz Tschebysheff filter (same for all input and output channels apart from a gain factor):

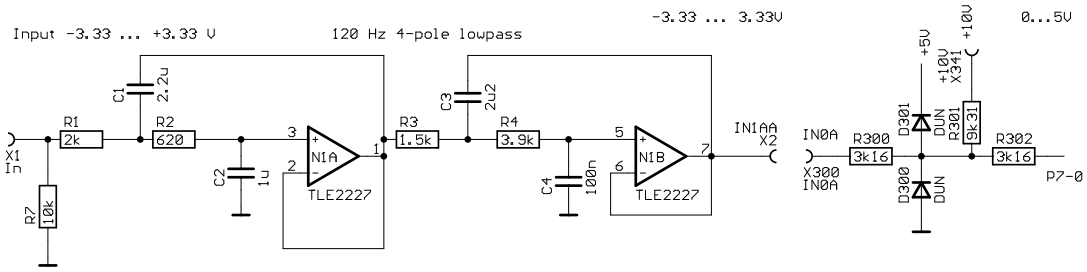



The filters are realized as 2-stage Sallen-Key circuits (see below). Furthermore, all A/D inputs of the 3048 are protected with a resistor-diode network. Channels 0...5 have an additional resistor to the stabilized 10 V reference which is used for level-shifting.

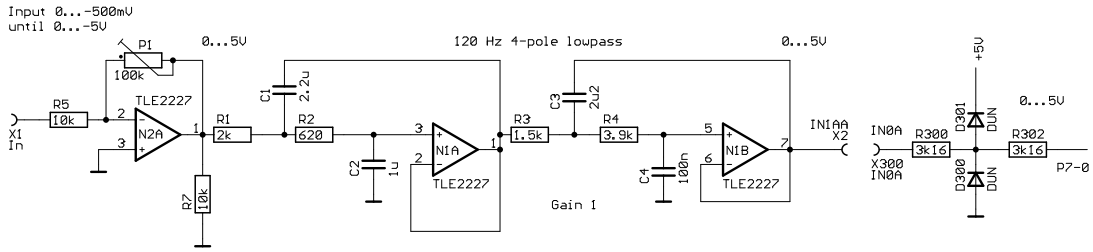
The following diagram shows the input circuit for channels 0...3:



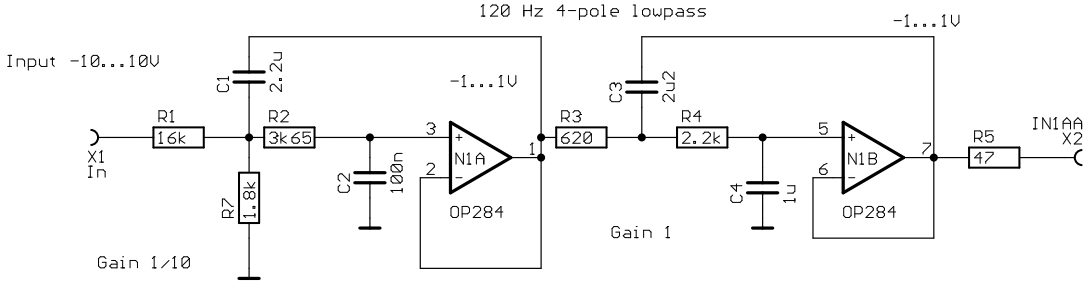

The following diagram shows the input circuit for channels 4 and 5:



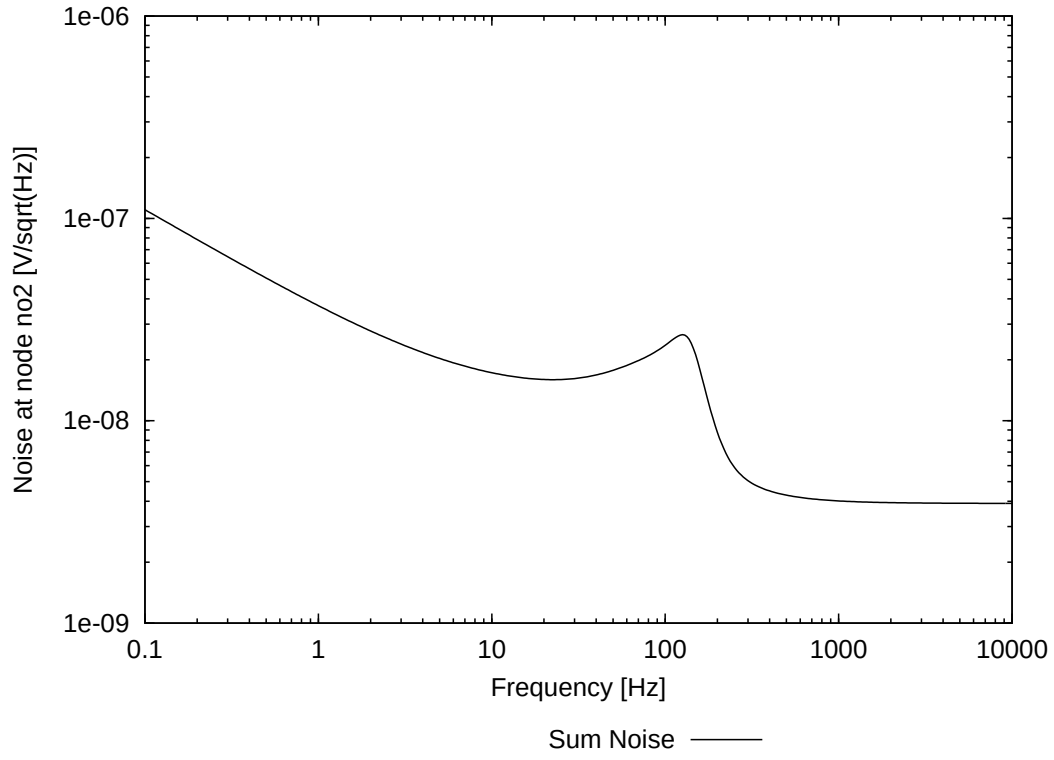

The following diagram shows the input circuit for channels 6 and 7:




The output signal comes from 2 DAC8420 D/A converters which operate with stabilized -10 V and +10 V reference voltages. Hence the output range is -10 V. . . +10 V. Since the required range is only -1 V. . . +1 V, the output filters have a fixed gain of 1/10. Apart from the gain, they have the same frequency response as the input filters. The following diagram shows the output filter circuit for all channels:




The following diagram shows the output noise spectral density of the output filters. Above 200 Hz, it is dominated by the voltage noise of the final op-amp.



# 10 Digital Filters

We use recursive digital filters (IIR filters). Here some relevant formulae are summarized. A digital filter transforms an input sequence $x_n$, which is sampled with a sampling interval $T$ (in our case $T = 1\,\mathrm{ms}$), into an output sequence $y_n$ by the relationship

$$y_i = a_0\,x_i + a_1\,x_{i-1} + \cdots + a_n\,x_{i-n} - b_1\,y_{i-1} - b_2\,y_{i-2} - \cdots - b_m\,y_{i-m}. \tag{23}$$

The transfer function of this filter is given by

$$H(s) = \frac{a_0 + a_1 z^{-1} + \cdots + a_n z^{-n}}{1 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m}}, \tag{24}$$

where

$$z = \exp(sT), \quad s = i\omega = 2\pi i f. \tag{25}$$

Stability of the filter is determined only by the denominator of this equation. The filter is stable if and only if all complex roots of the polynomial equation

$$z^m + b_1 z^{m-1} + b_2 z^{m-2} + \cdots + b_m = 0 \tag{26}$$

are within the unit circle, i.e. $|z| < 1$.

For this application we investigate three types of digital filters:

**Perfect Integrators** These have the simple transfer function $H(s) = \alpha/s$, with gain proportional to $1/f$ and a constant $-90°$ phase shift for all frequencies. Their coeffcients can be directly computed (see Section 10.1 below).

**Damped Integrators** These are "stopped" integrators with the gain settling at some finite gain and the phase shift returning to zero for higher frequencies. Again their coeffcients can be directly computed (see Section 10.2 below).

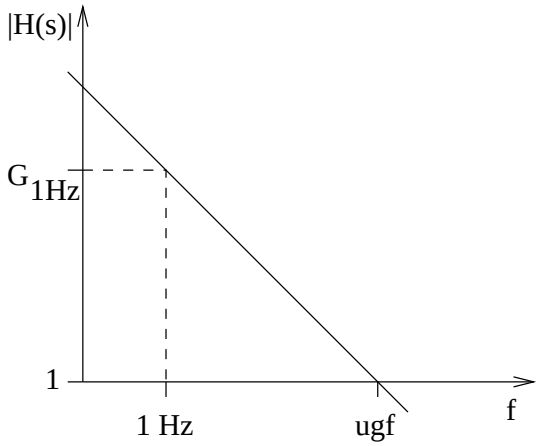
**General filters** In order to compensate actuator resonances etc., more complicated filters are necessary. They can be computed with LISO (see Section 11 below).

## 10.1 Perfect Integrators

The transfer function of a perfect integrator is given by

$$H(s) = \frac{\alpha}{s}, \tag{27}$$

and sketched here:




Some relationships (omitting the unit Hz):

$$G_{1\,\mathrm{Hz}} = \frac{\alpha}{2\pi}, \qquad \alpha = G_{1\,\mathrm{Hz}} \cdot 2\pi, \tag{28}$$

$$\mathrm{ugf} = \frac{\alpha}{2\pi}, \qquad \alpha = \mathrm{ugf} \cdot 2\pi. \tag{29}$$

The filter is realized by the recursive relation

$$y_i = a_0 x_i + y_{i-1}, \tag{30}$$

which has the transfer function

$$H(s) = \frac{a_0}{1 - z^{-1}} = \frac{a_0}{1 - \exp(-sT)} \approx \frac{a_0}{1 - (1 - sT)} = \frac{a_0}{sT}. \tag{31}$$

The approximation is valid for small frequencies $f \ll 1/T$. We see that

$$a_0 = \alpha T, \tag{32}$$

and hence

$$a_0 = 2\pi\, T\, G_{1\,\mathrm{Hz}} = 2\pi\, T\, \mathrm{ugf}. \tag{33}$$

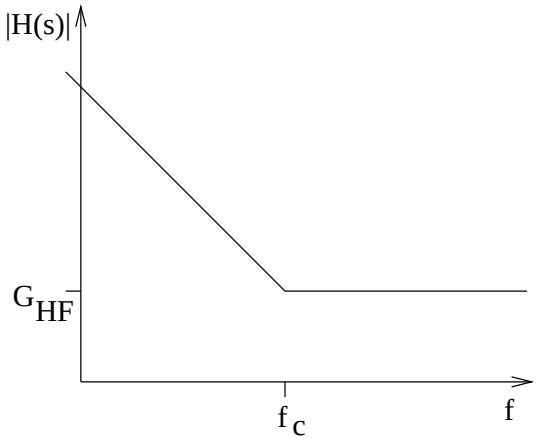
Thanks to the simple relationship Eq. (30) the filter can be realized efficiently in software, and slew-rate limiting is straightforward by limiting the increment $a_0 x$, which is equivalent to temporarily limiting the gain.

### 10.2 Damped Integrators

The transfer function of a damped integrator is given by

$$H(s) = G_\infty \left(1 + \frac{2\pi\, f_c}{s}\right) \tag{34}$$

and shown here:




It is characterized by the gain for high frequenices, $G_\infty$, and the corner frequency $f_c$, and can be realized by the recursive relation

$$y_i = a_0\, x_i + a_1\, x_{i-1} + y_{i-1}, \tag{35}$$

which has the transfer function

$$H(s) = \frac{a_0 + a_1 z^{-1}}{1 - z^{-1}} = \frac{a_0 + a_1 \exp(-sT)}{1 - \exp(-sT)}. \tag{36}$$

The approximation for small frequencies by expanding the exponential function yields:

$$H(s) \approx \frac{a_0 + a_1 - a_1 sT}{sT} = -a_1 + \frac{a_0 + a_1}{sT}. \tag{37}$$

By comparing equations (34) and (37) we find

$$a_1 = -G_\infty, \qquad a_0 = G_\infty(1 + 2\pi\, T\, f_c). \tag{38}$$

Again this filter can be implemented efficiently in software, and slew-rate limiting can be realized by limiting the increment $a_0\, x_i + a_1\, x_{i-1}$. Of course the transfer function (which is defined only for linear systems) is corrupted in the case of slew-rate limiting.

# 11 Some notes on IIR filters on the H3048F

In this section I describe some attempts at realizing more complex IIR filters with the 3048F processor. Although theses attempts finally failed, the information may be useful for future designs.

## 11.1 Purpose

Looking at the transfer functions in Section 6 it is clear that without special filters the unity gain frequency (UGF) will be limited to 1 or 2 Hz for each loop due to

actuator resonances. Originally I had intended to obtain a UGF of around 20 Hz for better suppression of the natural motion of the mirrors. The transfer functions are well defined up to that frequency and show only a few "clean" resonances each, which need to be compensated with filters.

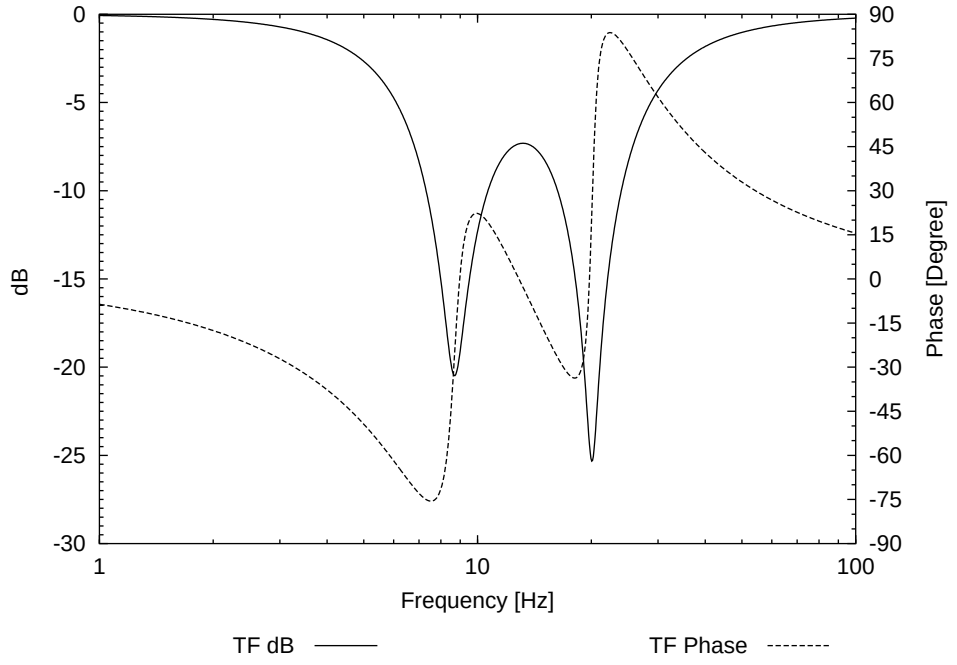
The plan was to use digital IIR filters inside the 3048F for that purpose. The main advantage of doing so is that the final output signal is under control of the microprocessor, enabling slew-rate limiting, a perfect "hold" function etc. If, on the other hand, the compensation filters are realized as analog external filters, the microprocessor has no way to achieve these desirable features.

In the following I will use the input mirror pitch PZT transfer function from Section 6.3 as example. It has two resonances at 8.7 Hz and 20 Hz with a $Q$ of 9.3 and 16.3, respectively. The sampling frequency is assumed to be

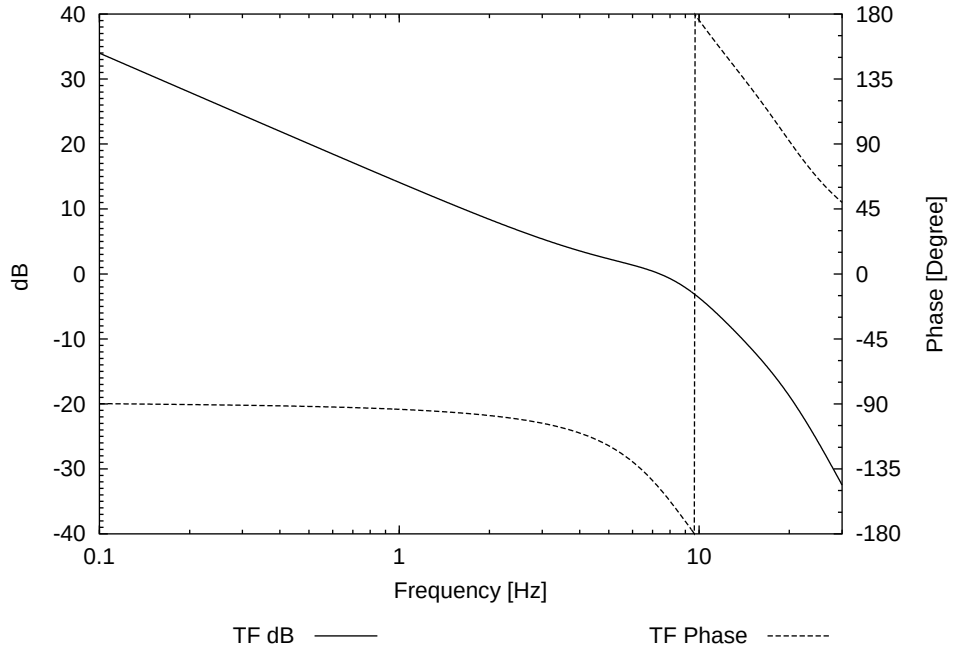
$$f_{\mathrm{samp}} = 1000\,\mathrm{Hz}. \tag{39}$$

The first idea to compensate these resonances is to introduce zeroes at these frequencies. However, without introducing new poles the required filter transfer function would rise as $f^4$ above 20 Hz, which introduces considerable noise and dynamic range problems. Hence the filter I discuss here introduces new poles of $Q = 1$ at the same frequencies of the old poles, i.e. effectively reducing the $Q$ of the poles to 1 without changing their frequency. The following plot shows the filter transfer function. It was generated by LISO with

```
zero 8.688 9.34
zero 20.08 16.34
pole 8.688 1
pole 20.08 1
```



With such a compensation filter, a UGF of around 7 Hz could have been achieved by using an integrator active between DC and 10 Hz:




## 11.2 Design of IIR filters with LISO

In order to design these IIR filters, LISO was extended (version 1.81). The following input file (called `m1yloop.fil`)

```
zero 8.688 9.34
zero 20.08  16.34
pole 8.688 1
pole 20.08  1
freq log .5 200 300
iir 1000 start 5 5
```

produces initial values for the filter (in a file called `m1yloop_iir.fil`) which are then refined by running LISO on that file, giving the following coefficients:

$$a_0 = 0.80903621 \tag{40}$$
$$a_1 = -3.744904 \tag{41}$$
$$a_2 = 6.9131052 \tag{42}$$
$$a_3 = -6.3548303 \tag{43}$$
$$a_4 = 2.904875 \tag{44}$$
$$a_5 = -0.5272688 \tag{45}$$
$$b_0 = 1.0 \tag{46}$$
$$b_1 = -4.5362002 \tag{47}$$
$$b_2 = 8.2212406 \tag{48}$$
$$b_3 = -7.4378974 \tag{49}$$
$$b_4 = 3.3572789 \tag{50}$$
$$b_5 = -0.6044083 \tag{51}$$

and a transfer function that is reasonably close to the desired one. However, it turns out that the required precision of the coefficients is of the order of $10^{-6} \ldots 10^{-7}$, which is hard to achieve with integer arithmetic. It was found that the filter can be realized more easily by cascading two filters that compensate one pole each. Therefore in the following the compensation filter that handles the first resonance at 8.7 Hz will be investigated. It can be designed by running LISO with the following input file (called `m1yl1.fil`)

```
zero 8.688 9.34
pole 8.688 1
freq log 1 100 200
iir 1000 start
```

and then running the automatically produced file (called `m1yl1_iir.fil`). This second run of LISO uses the fitting algorithm to refine the analytical initial values, requiring stability of the filter as a constraint. The coefficients are then rounded to 16-bit integers, trying various denominators and rounding up or down, and finally selecting those integer coefficients that produce the best transfer function. The result is the following filter:

$$y_i = \frac{8216\,x_i - 16360\,x_{i-1} + 8168\,x_{i-2}}{8416} - \frac{-22017\,y_{i-1} + 10723\,y_{i-2}}{11326}. \tag{52}$$

The computed transfer function of that filter (by Equation (24)) is close to the desired one, and it is also stable. After arriving at this point I had believed that the required filters *can* be realized in software, after all.

### 11.3 Limitations of the 3048F

Experience from earlier projects and preliminary tests of this circuit and software revealed the following limitations of the 3048F for the present application:

Arithmetic with sufficient speed is limited to (signed) integer variables. Although the gnu C compiler provides a floating point library (with 32-bit single precision `float` variables and all usual functions), these are far too slow.

Furthermore, multiply and divide operations are limited to the hardware `MULXS.W` and `DIVXS.W` instructions, which provide a $16 \times 16 \rightarrow 32$ bit multiplication and a $32/16 \rightarrow 16$ bit division in 24 cyles (1.5 $\mu$sec) each.

Adding and subtracting 32-bit numbers is no problem (2 cyles = 0.125 $\mu$sec for a register–register operation). However, multiply and divide operations for 32-bit numbers require many instructions and need at least 10 $\mu$sec for multiplication and 50 $\mu$sec for division.

In order to use the $32/16 \rightarrow 16$ bit `DIVXS.W` instruction, it is necessary to provide assembler code for the divide operation as follows:

```
inline int
mydiv (long a, int b)
{
  long result;
   asm ("divxs.w %1,%0": "=r" (result): "r" (b), "0" (a):"cc");
  return (int) result;
}

...
c=mydiv(a,b);
...
```

Otherwise the compiler will issue a subroutine call to the full 32-bit division routine.

Hence the filter operations are limited to 16-bit filter coefficients and 16-bit input and output values. Intermediate products and accumulators can use 32 bit, if only add, subtract or $32/16 \rightarrow 16$ bit division operations are used.

The example (52) above was designed taking into account these limitations.

Real timing data was obtained by programming the prototype, toggling digital output bits and measuring the time with an oscilloscope. The results are (for 6 input channels A/D):

**Getting** the data and subtracting the offset: 10 $\mu$sec for 6 channels.

**Matrix** operations on 16-bit integers to seperate the error signals: 118 $\mu$sec for 6 channels.

**DAC** output of 6 channels: 108 $\mu$sec for 6 channels.

**IIR filter** such as Equation (52): 50 $\mu$sec for **one channel**

The A/D converter was running continuously during these measurements, updating 16-bit accumulators via an interrupt handler. Once per sampling interval (1000 $\mu$sec) the accumulators are divided by the number of conversions and made available as input to the main program. This operation consumes around 10 . . . 20% of the CPU time.

The above results show that with filters of the type (52), operation with 6 channels at 1000 Hz should be possible. On the other hand, more complicated filters (using 32 bit or floating point variables) are out of the question.

The proposed sampling frequency of 1000 Hz may seem high for 20 Hz UGF. However, the phase shifts introduced by the the antialiasing filters (see Section 9) and those inherent in the digital filter sampling process prohibit much slower sampling.

### 11.4 Performance of the IIR filter

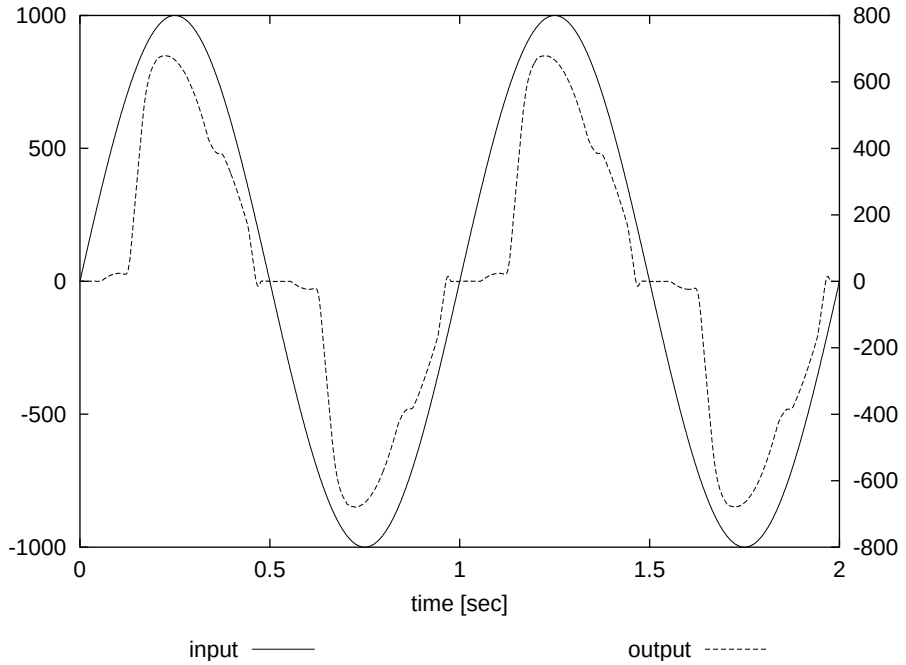
The IIR filter (52) was implemented and tested in the prototype. At first the performance seemed satisfactory, i.e. a transfer function measured with the spectrum analyzer corresponded to the prediction. More detailed tests, however, revealed a severe problem:

**The filter does not react well to small input signals** (e.g. a few % of the maximal amplitude).

To analyze this problem, LISO was further enhanced to simulate the integer operations of the filter. The following operations are done: For all interesting frequencies, first the maximum amplitude is determined that does not cause overflow in any intermediate result. For the example discussed here, that amplitude was 23675.[3] Next, the transfer function is determined by letting the filter operate (with integer operations) on an input sine wave, and 'demodulating' the output by multiplication with $\sin(\omega t)$ and $\cos(\omega t)$ (after allowing for settling for a few periods of the input signal). The input amplitude is then reduced in steps until the transfer function deviates by 3 dB (taking into account the phase also) from the initial transfer function that was found at big amplitudes. The smallest amplitude thus found is considered the lower limit of permissible input signals.

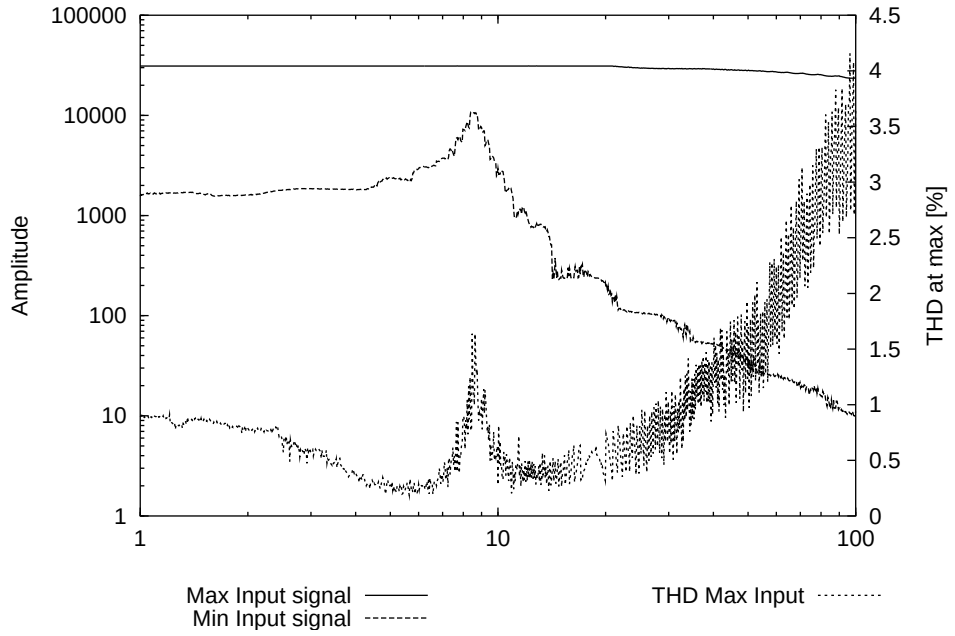
For the present example, this lower limit turned out to be 1560 at 1 Hz (i.e. well below the 8.7 Hz resonance). Here is what happens to a 1 Hz signal of amplitude 1000 (i.e. 4.2% of the full range) in the time domain:

[3] The limit was caused by the second fraction of Equation (52) for 100/,Hz input signals.




Clearly there is considerable crossover distortion. If the input signal amplitude is further decreased the response deteriorates rapidly and soon vanishes alltogether.

The following plot shows the minimal and maximal input amplitudes for the filter (52):




The top curve (referred to the left $y$-axis) is the maximal signal that avoids overflow, the next curve (also referred to the left $y$-axis) is the minimal signal that yields a deviation of 3 dB in the transfer function, and the bottom curve (referred to the right $y$-axis) is the total harmonic distortion for a signal of maximal amplitude.

The cause of the problem lies in equation (52), shown again here in floating-point form:

$$y_i = 0.9762\,x_i - 1.9439\,x_{i-1} + 0.9705\,x_{i-2} + 1.9439\,y_{i-1} - 0.9467\,y_{i-2}. \tag{53}$$

The coefficients of $x_i$, $x_{i-1}$ and $x_{i-2}$ nearly cancel each other. Hence the transmission of a slow signal from the input to the output relies on a signal appearing in the output. For small amplitudes, that build-up of the output signal does not take place, and the filter collapses. This effect is illustrated in the following table:

| $t$ | in$*$ | out$*$ | in | out | $a_0$ | $a_1$ | $a_2$ | $b_1$ | $b_2$ |
|---|---|---|---|---|---|---|---|---|---|
| 0.000 | 0.000 | 0.000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.001 | 0.628 | 0.613 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.002 | 1.257 | 1.198 | 1 | 0 | 0 | -1 | 0 | 0 | 0 |
| 0.003 | 1.885 | 1.755 | 2 | 0 | 1 | -1 | 0 | 0 | 0 |
| 0.004 | 2.513 | 2.286 | 3 | 0 | 2 | -3 | 0 | 0 | 0 |
| 0.005 | 3.141 | 2.793 | 3 | 0 | 2 | -5 | 1 | 0 | 0 |
| 0.006 | 3.769 | 3.278 | 4 | 0 | 3 | -5 | 2 | 0 | 0 |
| 0.007 | 4.397 | 3.742 | 4 | 0 | 3 | -7 | 2 | 0 | 0 |
| 0.008 | 5.024 | 4.186 | 5 | 0 | 4 | -7 | 3 | 0 | 0 |
| 0.009 | 5.652 | 4.613 | 6 | 0 | 5 | -9 | 3 | 0 | 0 |
| 0.010 | 6.279 | 5.023 | 6 | 0 | 5 | -11 | 4 | 0 | 0 |
| 0.011 | 6.906 | 5.419 | 7 | 0 | 6 | -11 | 5 | 0 | 0 |
| 0.012 | 7.533 | 5.801 | 8 | 0 | 7 | -13 | 5 | 0 | 0 |
| 0.013 | 8.159 | 6.171 | 8 | 0 | 7 | -15 | 6 | 0 | 0 |
| 0.014 | 8.785 | 6.531 | 9 | 0 | 8 | -15 | 7 | 0 | 0 |
| 0.015 | 9.411 | 6.881 | 9 | 0 | 8 | -17 | 7 | 0 | 0 |
| 0.016 | 10.036 | 7.224 | 10 | 1 | 9 | -17 | 8 | 0 | 0 |
| 0.017 | 10.661 | 7.559 | 11 | 1 | 10 | -19 | 8 | 1 | 0 |
| 0.018 | 11.286 | 7.889 | 11 | 0 | 10 | -21 | 9 | 1 | 0 |
| 0.019 | 11.910 | 8.214 | 12 | 1 | 11 | -21 | 10 | 0 | 0 |
| 0.020 | 12.533 | 8.536 | 13 | 1 | 12 | -23 | 10 | 1 | 0 |
| 0.021 | 13.156 | 8.855 | 13 | 0 | 12 | -25 | 11 | 1 | 0 |
| 0.022 | 13.779 | 9.173 | 14 | 1 | 13 | -25 | 12 | 0 | 0 |
| 0.023 | 14.401 | 9.491 | 14 | 1 | 13 | -27 | 12 | 1 | 0 |
| 0.024 | 15.023 | 9.808 | 15 | 1 | 14 | -27 | 13 | 1 | 0 |
| 0.025 | 15.643 | 10.127 | 16 | 0 | 15 | -29 | 13 | 1 | 0 |
| 0.026 | 16.264 | 10.448 | 16 | 0 | 15 | -31 | 14 | 0 | 0 |
| 0.027 | 16.883 | 10.772 | 17 | 1 | 16 | -31 | 15 | 0 | 0 |
| 0.028 | 17.502 | 11.099 | 18 | 1 | 17 | -33 | 15 | 1 | 0 |
| 0.029 | 18.121 | 11.430 | 18 | 0 | 17 | -34 | 16 | 1 | 0 |
| 0.030 | 18.738 | 11.765 | 19 | 1 | 18 | -34 | 17 | 0 | 0 |

$t$ is the time in msec, "in$*$" a (floating point) input signal of 1 Hz and amplitude 100 and "out$*$" the output signal of the filter (53), implemented in floating point. The rest of the table refers to the filter (52), implemented in integer arithmetic. "in" is the input signal, "out" the output. The columns labelled "$a_0$" to "$b_2$" show the individual contributions to the output signal, as given in Equation (52). In the table, they don't

add up exactly, because the output signal is computed as

$$y_i = \frac{8216\,x_i - 16360\,x_{i-1} + 8168\,x_{i-2}}{8416} - \frac{-22017\,y_{i-1} + 10723\,y_{i-2}}{11326}, \tag{54}$$

whereas the table entries are shown as

$$\frac{8216\,x_i}{8416}, \quad \frac{-16360\,x_{i-1}}{8416}, \quad \frac{8168\,x_{i-2}}{8416}, \quad \frac{-22017\,y_{i-1}}{11326}, \quad \frac{10723\,y_{i-2}}{11326}, \tag{55}$$

respectively. Due to the near cancellation of the coefficients $a_0 + a_1 + a_2$, the output signal fails to build up in the presence of rounding errors. The DC gain, however, relies on an output signal building up, and hence the filter completely fails.

### 11.5 Attempts to improve the IIR filter

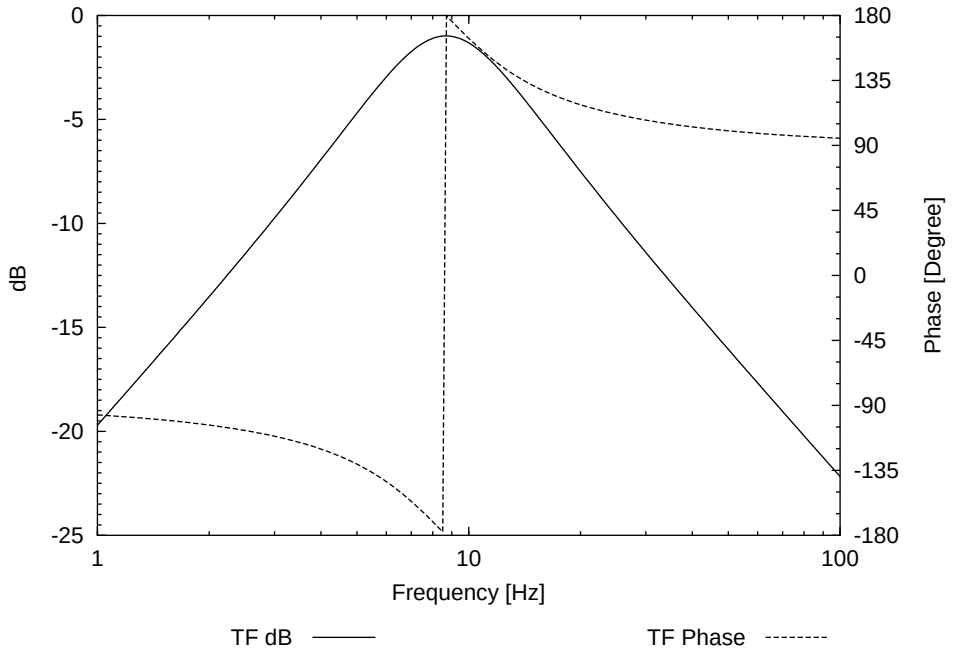
After the failure described in the previous section, I have tried to improve the filter by removing the DC gain, i.e. making an IIR filter that implements the transfer function

$$H'(s) = 1 - H(s), \tag{56}$$

such that the real output signal is found by

$$\text{output} = \text{input} + H'(s)(\text{input}). \tag{57}$$

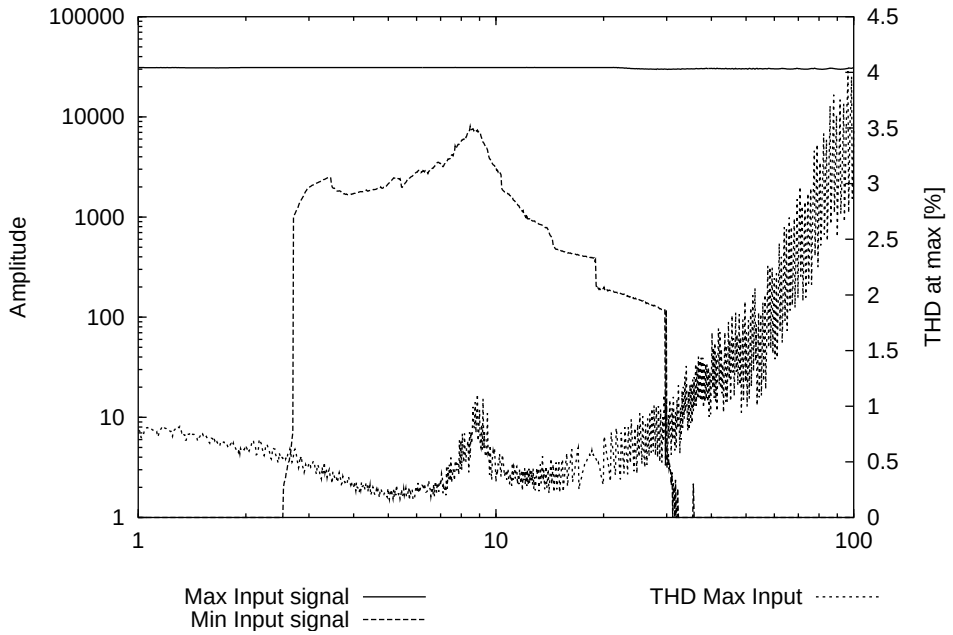
The hope was that a DC signal would be transferred directly to the output, bypassing the IIR filter, which would only become active for signals near the resonance frequency. The following plot shows the transfer function $H'(s)$:




LISO was extended once more to yield the following filter:

$$y_i = \frac{-778\,x_i + 0\,x_{i-1} + 778\,x_{i-2}}{32767} - \frac{-22053\,y_{i-1} + 10741\,y_{i-2}}{11345}. \tag{58}$$

The following plot shows again the minimal and maximal input signals for that filter (including the DC bypass), together with the harmonic distortion:




Although the performance for DC is now good, as expected, the performance remains poor for signals near the resonance. The sharp change of the minimal input signal is caused by the somewhat arbitrary limit of 3 dB deviation in the transfer function, which is not at all reached for signals far enough from the resonance.

I have concluded that **it is not possible to implement compensation filters with 16-bit integer arithmetic for the present project.**

Since the design of IIR filters itself is no problem, there is no fundamental problem in using them. For future designs I would try to find a suitable processor which has floating point arithmetic (e.g. a DSP or something like a 80486DX with 64/80-bit floating point coprocessor). It is, however, questionable, if such processors have all the convenient I/O features of the 3048F microcontroller, that are used for controlling the LCD, input switches, serial connection to the DAC etc.